US011244247B2

(12) **United States Patent**
Sainani et al.

(10) **Patent No.: US 11,244,247 B2**
(45) **Date of Patent: *Feb. 8, 2022**

(54) **FACILITATING CONCURRENT FORECASTING OF MULTIPLE TIME SERIES**

(71) Applicant: **SPLUNK Inc.**, San Francisco, CA (US)

(72) Inventors: **Manish Sainani**, Kirkland, WA (US); **Nghi Huu Nguyen**, Union City, CA (US); **Zidong Yang**, Millbrae, CA (US)

(73) Assignee: **Splunk Inc.**, San Francisco, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1 day.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **16/904,866**

(22) Filed: **Jun. 18, 2020**

(65) **Prior Publication Data**

US 2021/0042658 A1 Feb. 11, 2021

Related U.S. Application Data

(63) Continuation of application No. 15/143,335, filed on Apr. 29, 2016, now Pat. No. 10,726,354, which is a (Continued)

(51) **Int. Cl.**
G06N 20/00 (2019.01)
G06F 16/242 (2019.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06N 20/00*** (2019.01); ***G06F 16/22*** (2019.01); ***G06F 16/248*** (2019.01); ***G06F 16/2428*** (2019.01); ***G06F 16/2471*** (2019.01)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,631,392 B1 * 1/2014 Owen G06F 17/18 717/141
9,323,599 B1 4/2016 Iyer et al.
(Continued)

OTHER PUBLICATIONS

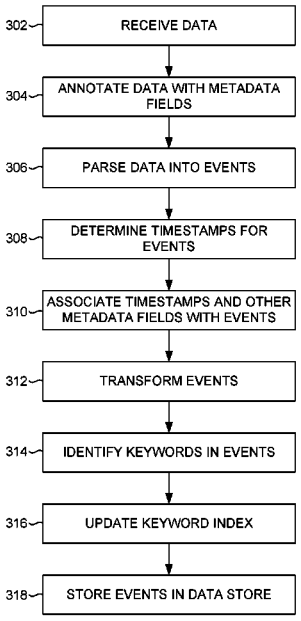
U.S. Appl. No. 15/143,335, filed Apr. 29, 2016, Which is: Granted.
(Continued)

Primary Examiner — Tracy C Chan
(74) *Attorney, Agent, or Firm* — Shook, Hardy & Bacon L.L.P.

(57) **ABSTRACT**

Embodiments of the present invention are directed to facilitating concurrent forecasting associating with multiple time series data sets. In accordance with aspects of the present disclosure, a request to perform a predictive analysis in association with multiple time series data sets is received. Thereafter, the request is parsed to identify each of the time series data sets to use in predictive analysis. For each time series data set, an object is initiated to perform the predictive analysis for the corresponding time series data set. Generally, the predictive analysis predicts expected outcomes based on the corresponding time series data set. Each object is concurrently executed to generate expected outcomes associated with the corresponding time series data set, and the expected outcomes associated with each of the corresponding time series data sets are provided for display.

20 Claims, 32 Drawing Sheets



Related U.S. Application Data

continuation-in-part of application No. 15/010,732, filed on Jan. 29, 2016.

(51) **Int. Cl.**

G06F 16/22	(2019.01)
G06F 16/2458	(2019.01)
G06F 16/248	(2019.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

2004/0068476	A1	4/2004	Provost et al.	
2005/0060693	A1	3/2005	Robison et al.	
2006/0173878	A1	8/2006	Bley	
2007/0198328	A1*	8/2007	Fuller	H04L 67/1097 711/170
2010/0280985	A1	11/2010	Duchon et al.	
2013/0024173	A1	1/2013	Brzezicki et al.	
2013/0103764	A1	4/2013	Verkasalo	
2013/0185232	A1	7/2013	Hochstein	
2015/0178834	A1	6/2015	Co et al.	
2016/0025020	A1*	1/2016	Hodzen	F02D 41/0007 701/103
2016/0062950	A1*	3/2016	Brodersen	G06F 17/18 702/181
2016/0063385	A1*	3/2016	Singh	G06F 17/18 706/50
2016/0217384	A1*	7/2016	Leonard	G06N 5/02
2017/0116524	A1*	4/2017	Hariharan	G06F 16/285

OTHER PUBLICATIONS

U.S. Appl. No. 15/010,732, filed Jan. 29, 2016, Which is: Pending.

Eckstein, B.A., "Evaluation of spline and weighted average interpolation algorithms," Computers & Geosciences, vol. 15, Issue 1, pp. 79-94 (1989).

Honaker, J., and King, G., "What to do about missing values in time-series cross-section data," American Journal of Political Science, vol. 54, No. 2, pp. 561-581 (Apr. 2010).

Moritz, S., et al., "Comparison of different methods for univariate time series imputation in R", Applications, arXiv:1510.03924, pp. 1-20 (2015).

Teegavarapu, R. S.V, and Chandramouli, V., "Improved weighting methods, deterministic and stochastic data-driven models for estimation of missing precipitation records," Journal of Hydrology, vol. 312, Issue 1-4, pp. 191-206 (2005).

* cited by examiner

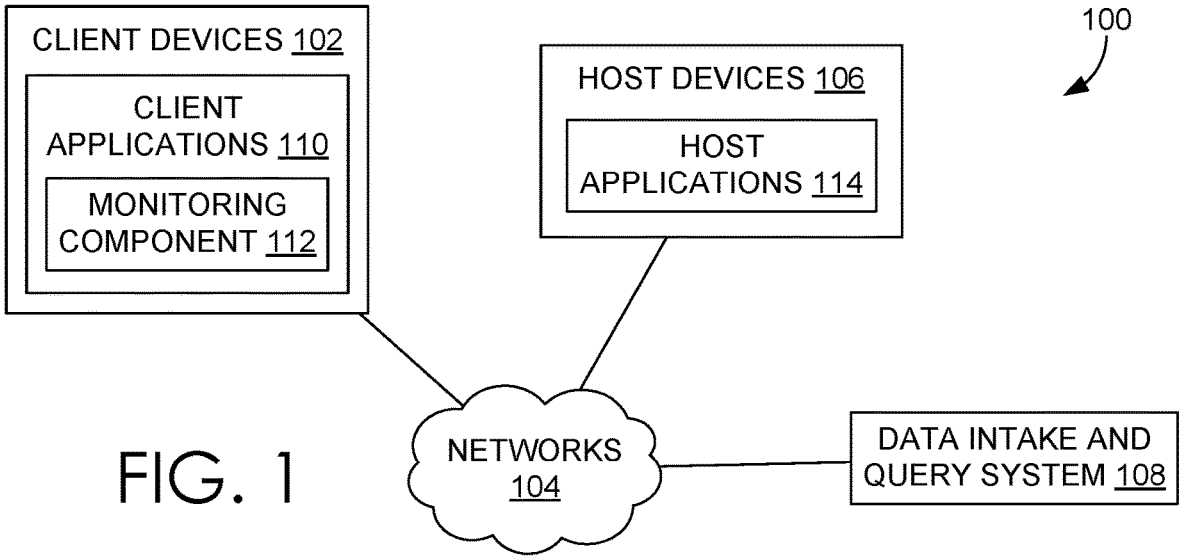

100
CLIENT DEVICES 102
CLIENT APPLICATIONS 110
MONITORING COMPONENT 112
HOST DEVICES 106
HOST APPLICATIONS 114
NETWORKS 104
DATA INTAKE AND QUERY SYSTEM 108

FIG. 1

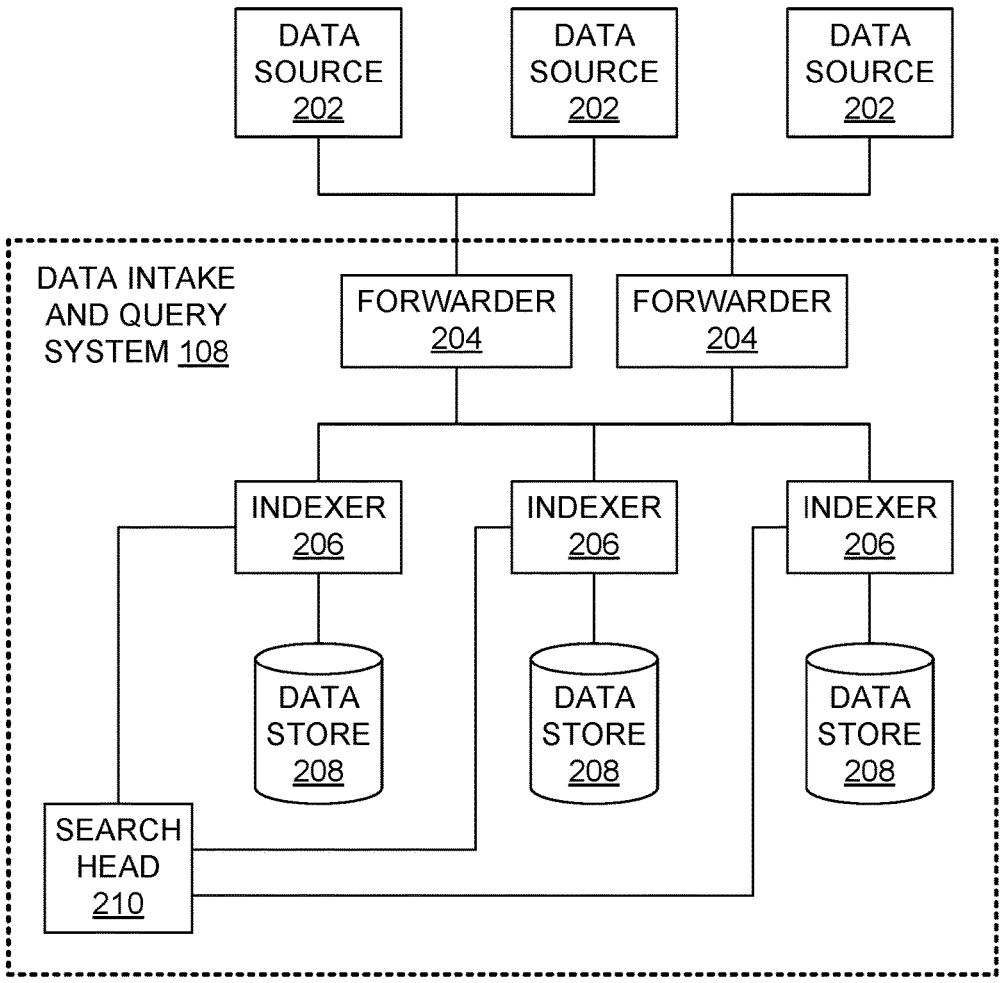

DATA SOURCE 202
DATA SOURCE 202
DATA SOURCE 202
DATA INTAKE AND QUERY SYSTEM 108
FORWARDER 204
FORWARDER 204
INDEXER 206
INDEXER 206
INDEXER 206
DATA STORE 208
DATA STORE 208
DATA STORE 208
SEARCH HEAD 210

FIG. 2

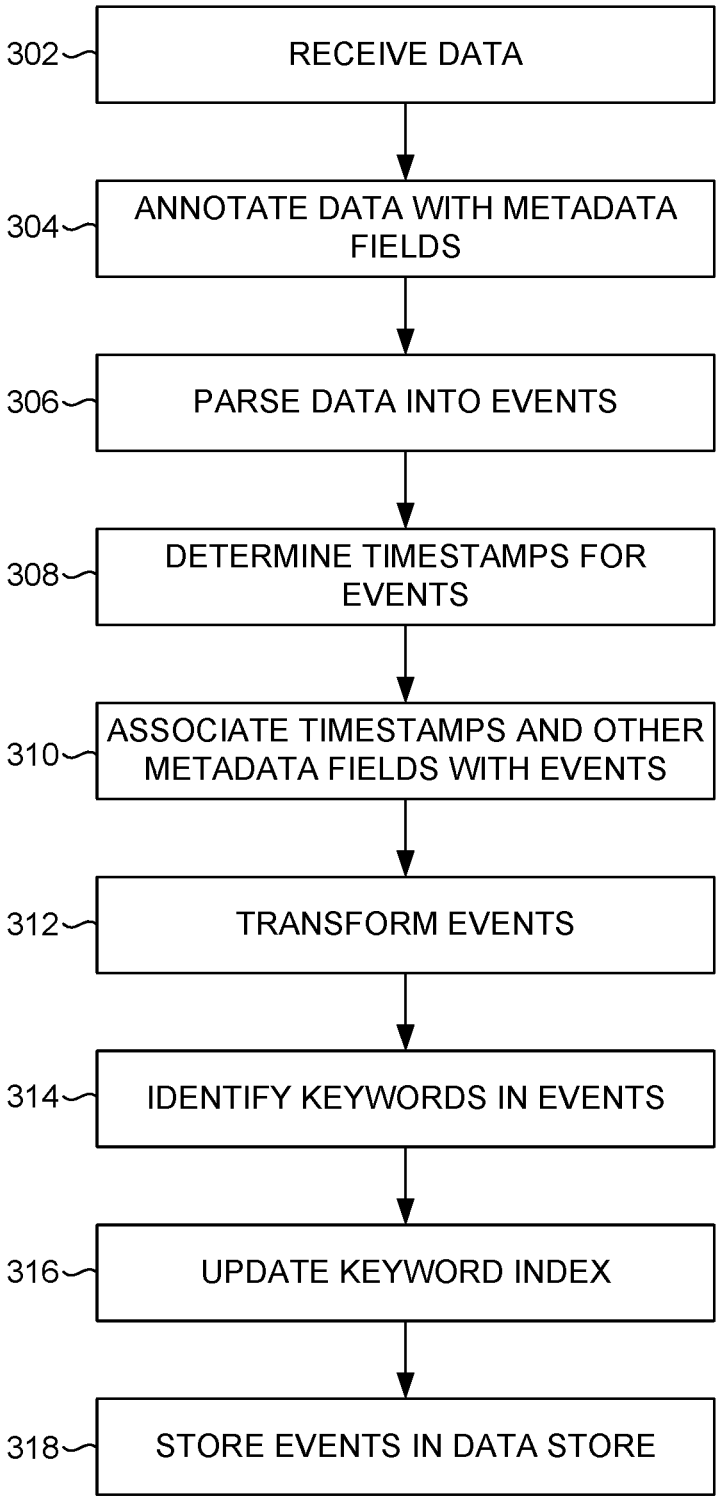

302
RECEIVE DATA
304
ANNOTATE DATA WITH METADATA FIELDS
306
PARSE DATA INTO EVENTS
308
DETERMINE TIMESTAMPS FOR EVENTS
310
ASSOCIATE TIMESTAMPS AND OTHER METADATA FIELDS WITH EVENTS
312
TRANSFORM EVENTS
314
IDENTIFY KEYWORDS IN EVENTS
316
UPDATE KEYWORD INDEX
318
STORE EVENTS IN DATA STORE

FIG. 3

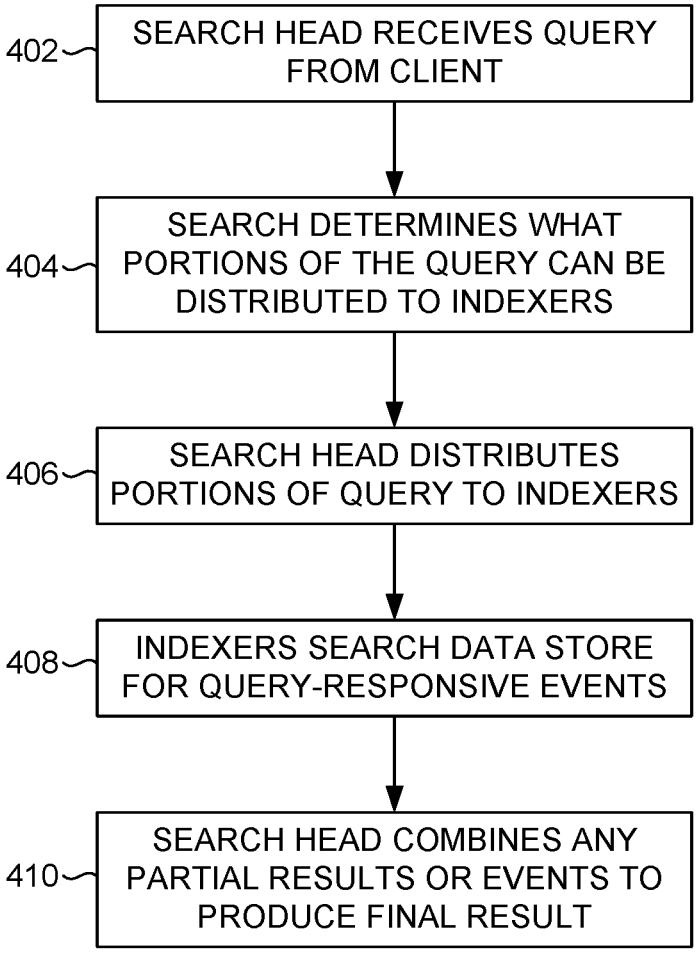

402
SEARCH HEAD RECEIVES QUERY FROM CLIENT
404
SEARCH DETERMINES WHAT PORTIONS OF THE QUERY CAN BE DISTRIBUTED TO INDEXERS
406
SEARCH HEAD DISTRIBUTES PORTIONS OF QUERY TO INDEXERS
408
INDEXERS SEARCH DATA STORE FOR QUERY-RESPONSIVE EVENTS
410
SEARCH HEAD COMBINES ANY PARTIAL RESULTS OR EVENTS TO PRODUCE FINAL RESULT

FIG. 4

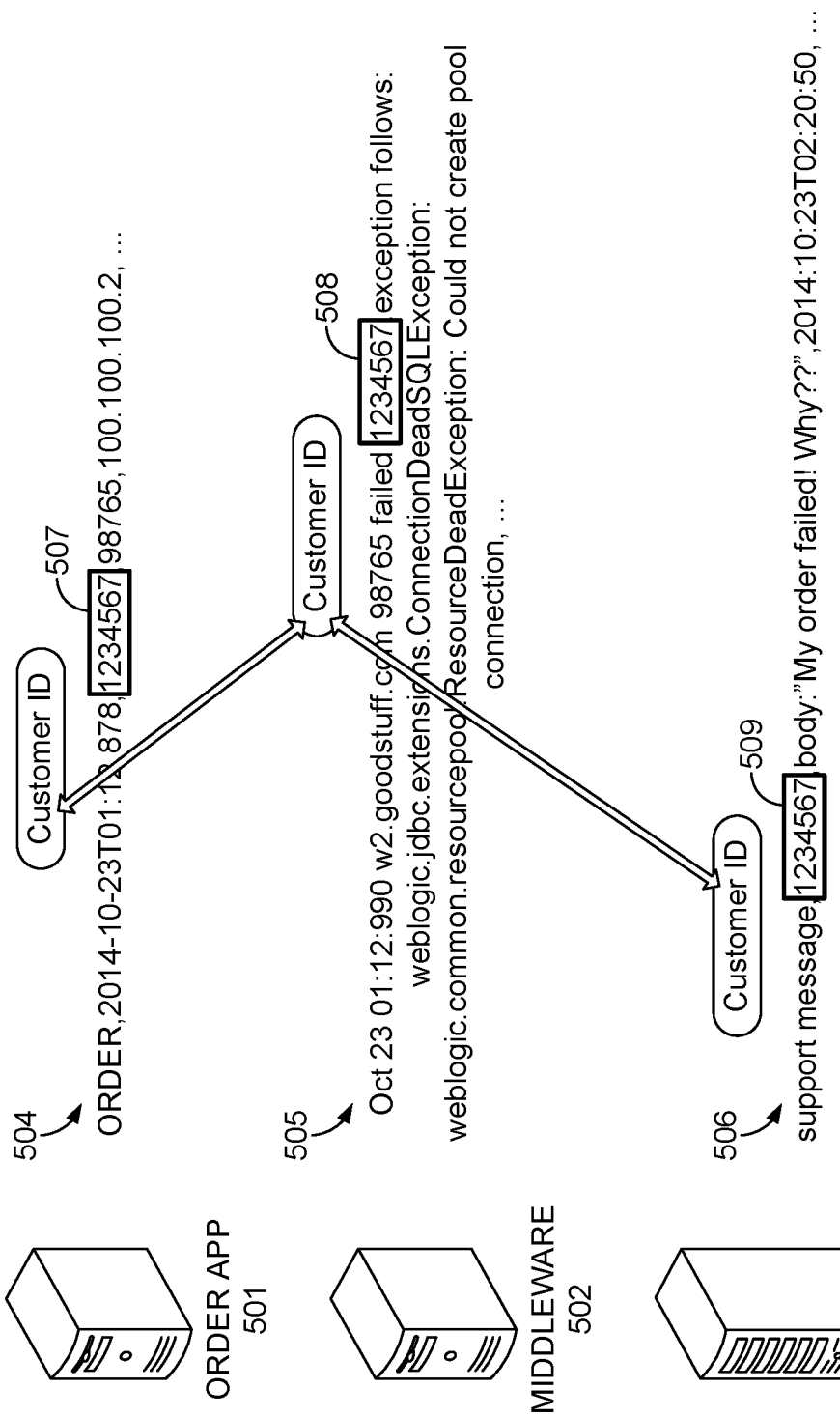

ORDER APP
501
MIDDLEWARE
502
SUPPORT
SERVER
503
504
ORDER,2014-10-23T01:12:878,1234567,98765,100.100.100.2, ...
Customer ID
507
505
Oct 23 01:12:990 w2.goodstuff.com 98765 failed 1234567 exception follows:
weblogic.jdbc.extensions.ConnectionDeadSQLException:
weblogic.common.resourcepool.ResourceDeadException: Could not create pool
connection, ...
Customer ID
508
506
support message,1234567,body:"My order failed! Why??",2014:10:23T02:20:50, ...
Customer ID
509

FIG. 5

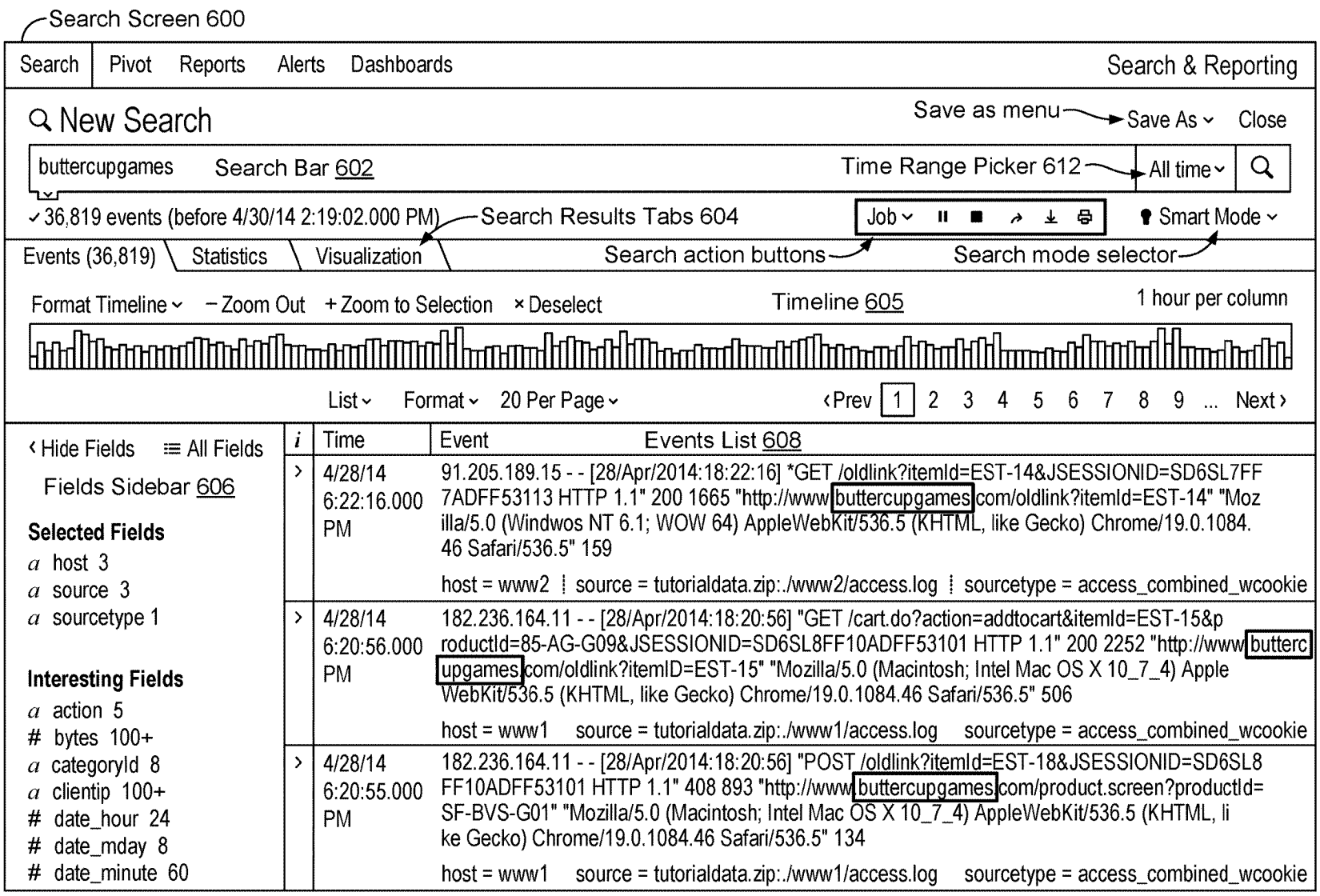

Search Screen 600
Search
Pivot
Reports
Alerts
Dashboards
Search & Reporting
New Search
Save as menu
Save As
Close
buttercupgames
Search Bar 602
Time Range Picker 612
All time
36,819 events (before 4/30/14 2:19:02.000 PM)
Search Results Tabs 604
Job
Smart Mode
Events (36,819)
Statistics
Visualization
Search action buttons
Search mode selector
Format Timeline
Zoom Out
Zoom to Selection
Deselect
Timeline 605
1 hour per column
List
Format
20 Per Page
Prev 1 2 3 4 5 6 7 8 9 ... Next
Hide Fields
All Fields
Fields Sidebar 606
Selected Fields
a host 3
a source 3
a sourcetype 1
Interesting Fields
a action 5
bytes 100+
a categoryId 8
a clientip 100+
date_hour 24
date_mday 8
date_minute 60
i
Time
Event
Events List 608
4/28/14 6:22:16.000 PM
91.205.189.15 - - [28/Apr/2014:18:22:16] *GET /oldlink?itemId=EST-14&JSESSIONID=SD6SL7FF7ADFF53113 HTTP 1.1" 200 1665 "http://www.buttercupgames.com/oldlink?itemId=EST-14" "Mozilla/5.0 (Windwos NT 6.1; WOW 64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 159
host = www2 | source = tutorialdata.zip:./www2/access.log | sourcetype = access_combined_wcookie
4/28/14 6:20:56.000 PM
182.236.164.11 - - [28/Apr/2014:18:20:56] "GET /cart.do?action=addtocart&itemId=EST-15&productId=85-AG-G09&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 200 2252 "http://www.buttercupgames.com/oldlink?itemID=EST-15" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 506
host = www1 source = tutorialdata.zip:./www1/access.log sourcetype = access_combined_wcookie
4/28/14 6:20:55.000 PM
182.236.164.11 - - [28/Apr/2014:18:20:56] "POST /oldlink?itemId=EST-18&JSESSIONID=SD6SL8FF10ADFF53101 HTTP 1.1" 408 893 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-G01" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 134
host = www1 source = tutorialdata.zip:./www1/access.log sourcetype = access_combined_wcookie

FIG. 6A

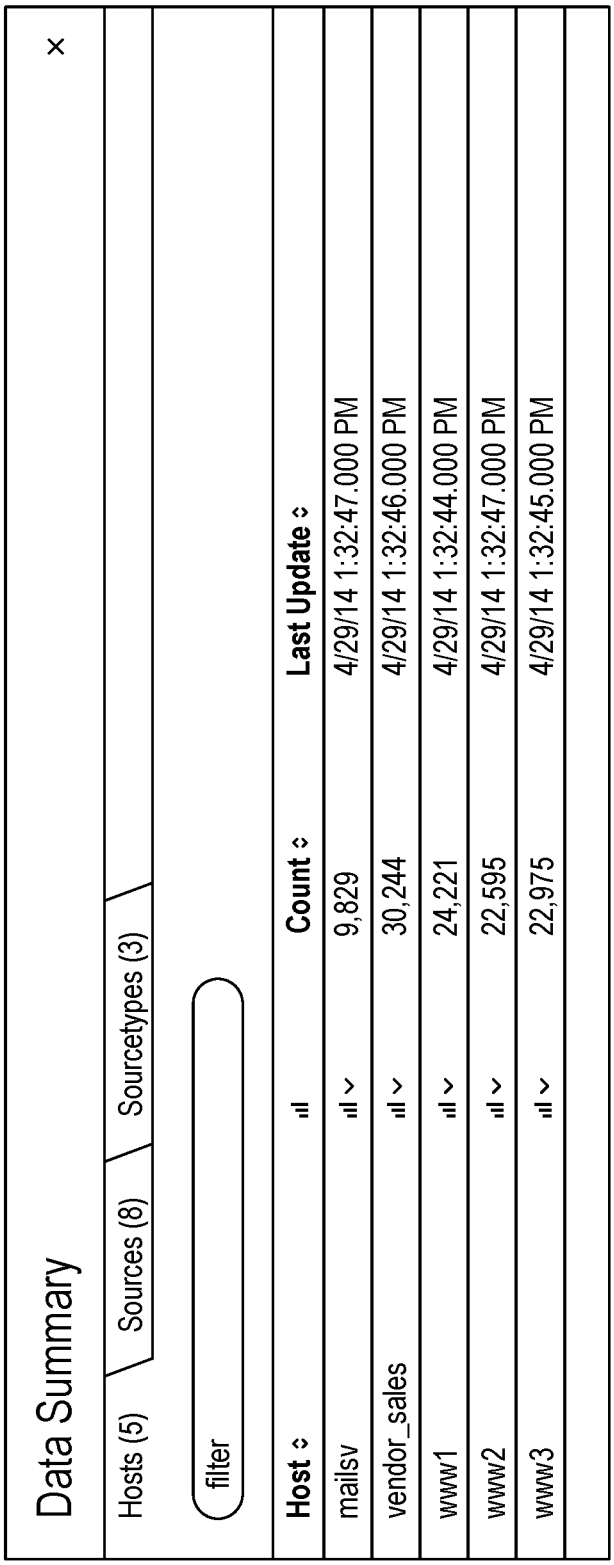

Data Summary
×
Hosts (5)
Sources (8)
Sourcetypes (3)
filter
Host
Count
Last Update
mailsv
9,829
4/29/14 1:32:47.000 PM
vendor_sales
30,244
4/29/14 1:32:46.000 PM
www1
24,221
4/29/14 1:32:44.000 PM
www2
22,595
4/29/14 1:32:47.000 PM
www3
22,975
4/29/14 1:32:45.000 PM

FIG. 6B

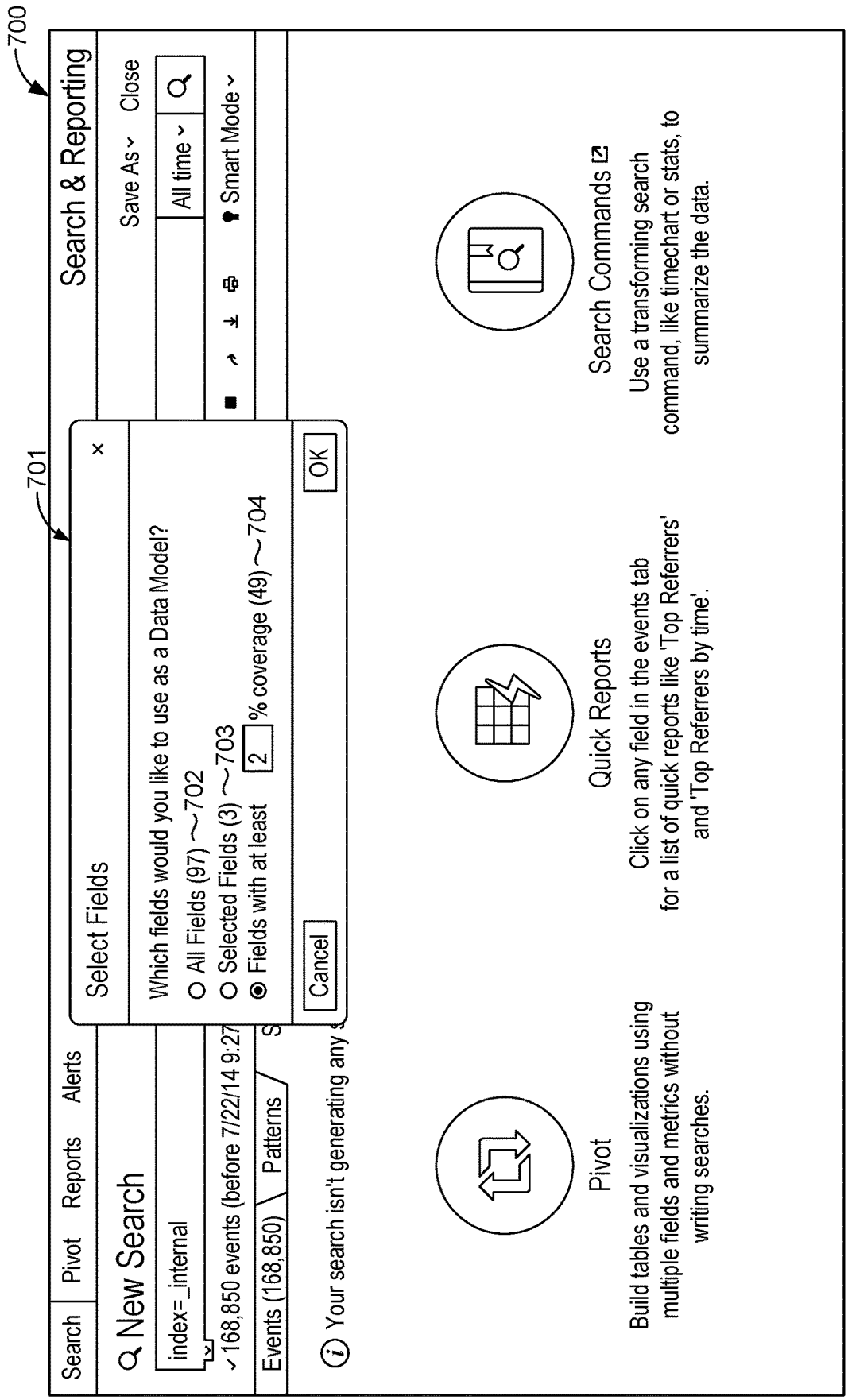

700
701
Search & Reporting
Search
Pivot
Reports
Alerts
Save As
Close
New Search
index=_internal
All time
Smart Mode
168,850 events (before 7/22/14 9:27
Events (168,850)
Patterns
Your search isn't generating any
Select Fields
Which fields would you like to use as a Data Model?
All Fields (97) 702
Selected Fields (3) 703
Fields with at least 2 % coverage (49) 704
Cancel
OK
Pivot
Build tables and visualizations using multiple fields and metrics without writing searches.
Quick Reports
Click on any field in the events tab for a list of quick reports like 'Top Referrers' and 'Top Referrers by time'.
Search Commands
Use a transforming search command, like timechart or stats, to summarize the data.

FIG. 7A

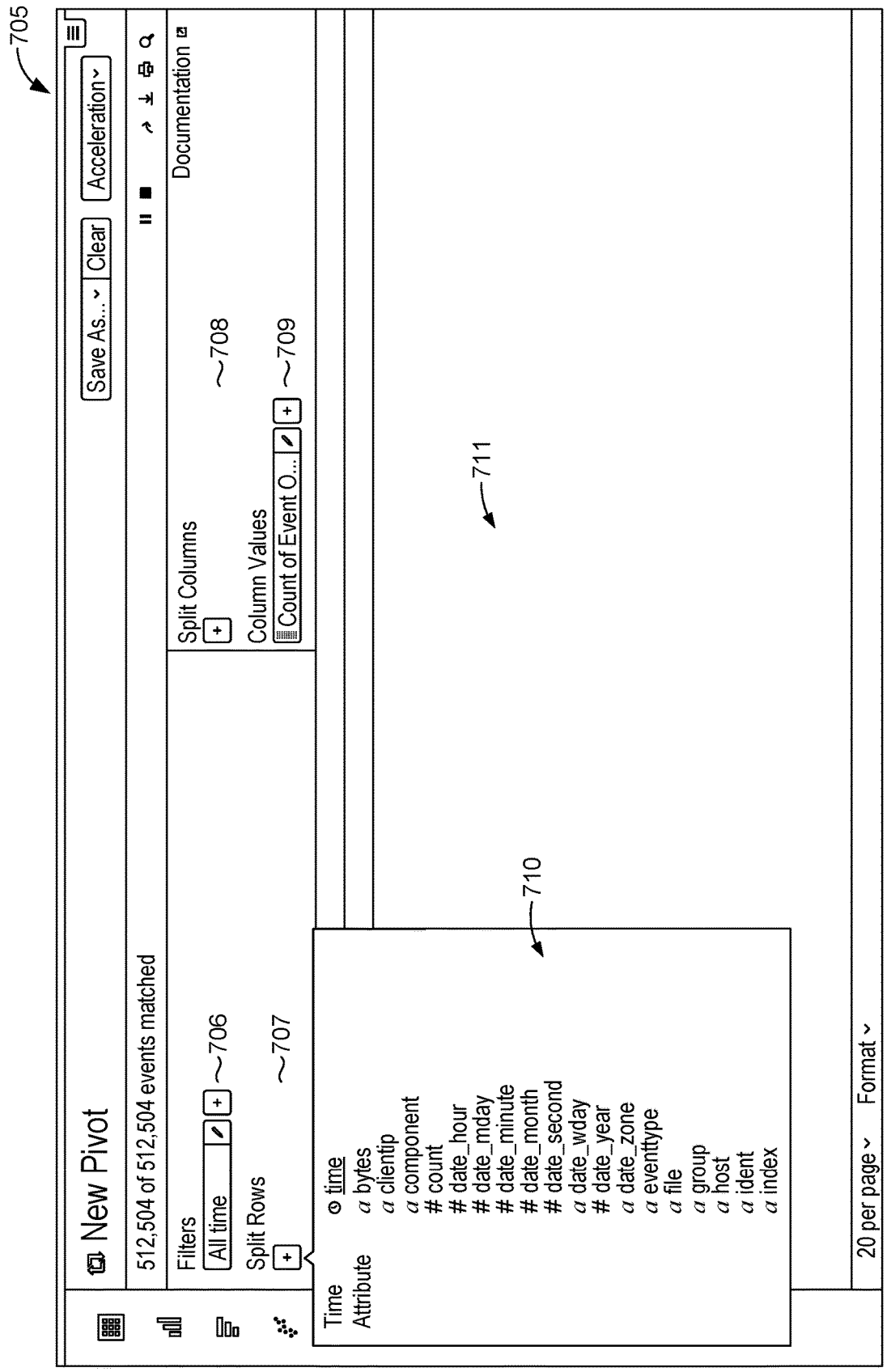

705
New Pivot
Save As...
Clear
Acceleration
512,504 of 512,504 events matched
Documentation
Filters
All time
706
Split Rows
707
Split Columns
708
Column Values
Count of Event O...
709
711
710
Time
Attribute
time
bytes
clientip
component
count
date_hour
date_mday
date_minute
date_month
date_second
date_wday
date_year
date_zone
eventtype
file
group
host
ident
index
20 per page
Format

FIG. 7B

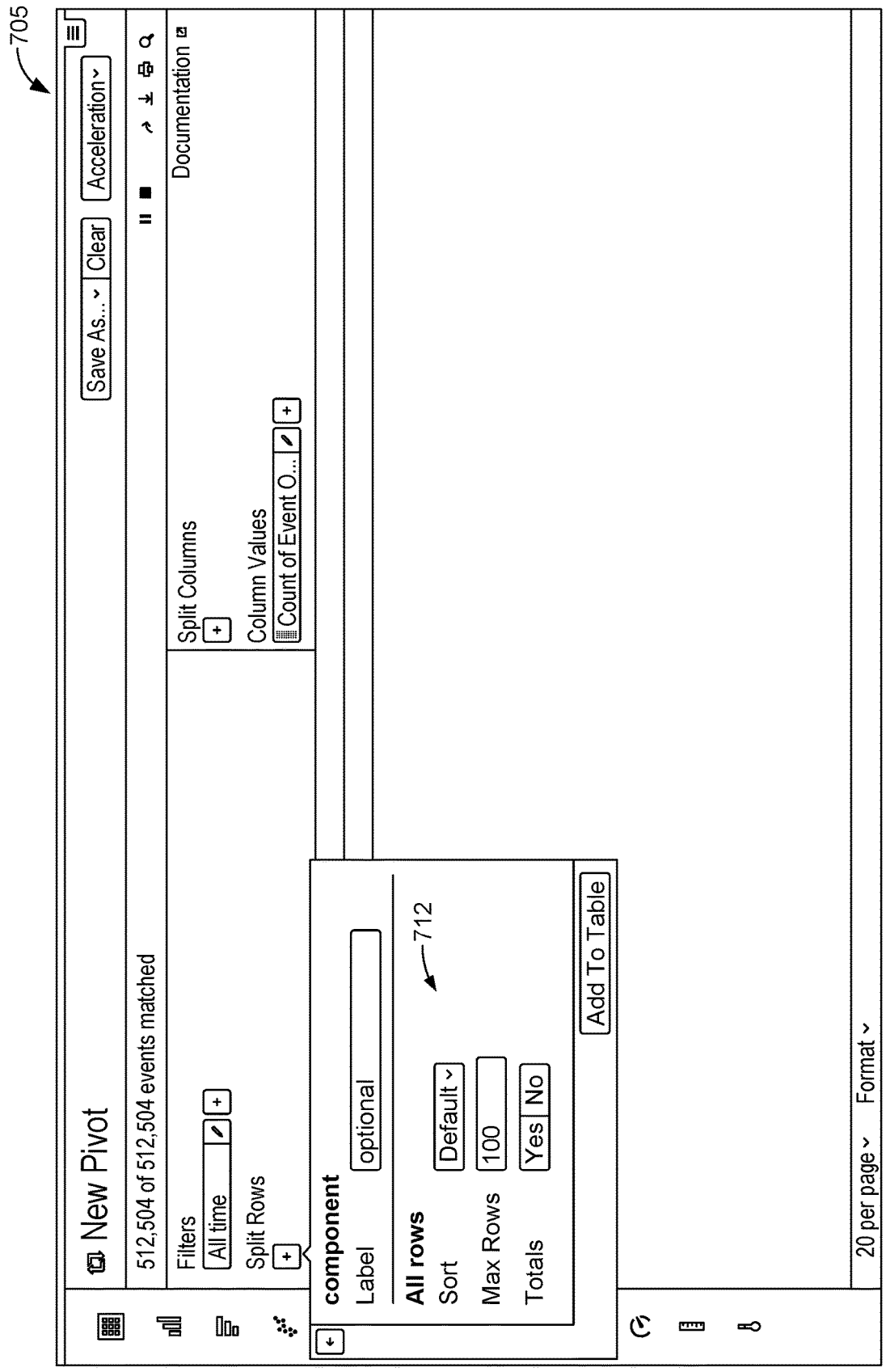

705
New Pivot
Save As...
Clear
Acceleration
512,504 of 512,504 events matched
Documentation
Filters
All time
Split Rows
Split Columns
Column Values
Count of Event O...
component
Label
optional
All rows
Sort
Default
Max Rows
100
Totals
Yes
No
712
Add To Table
20 per page
Format

FIG. 7C

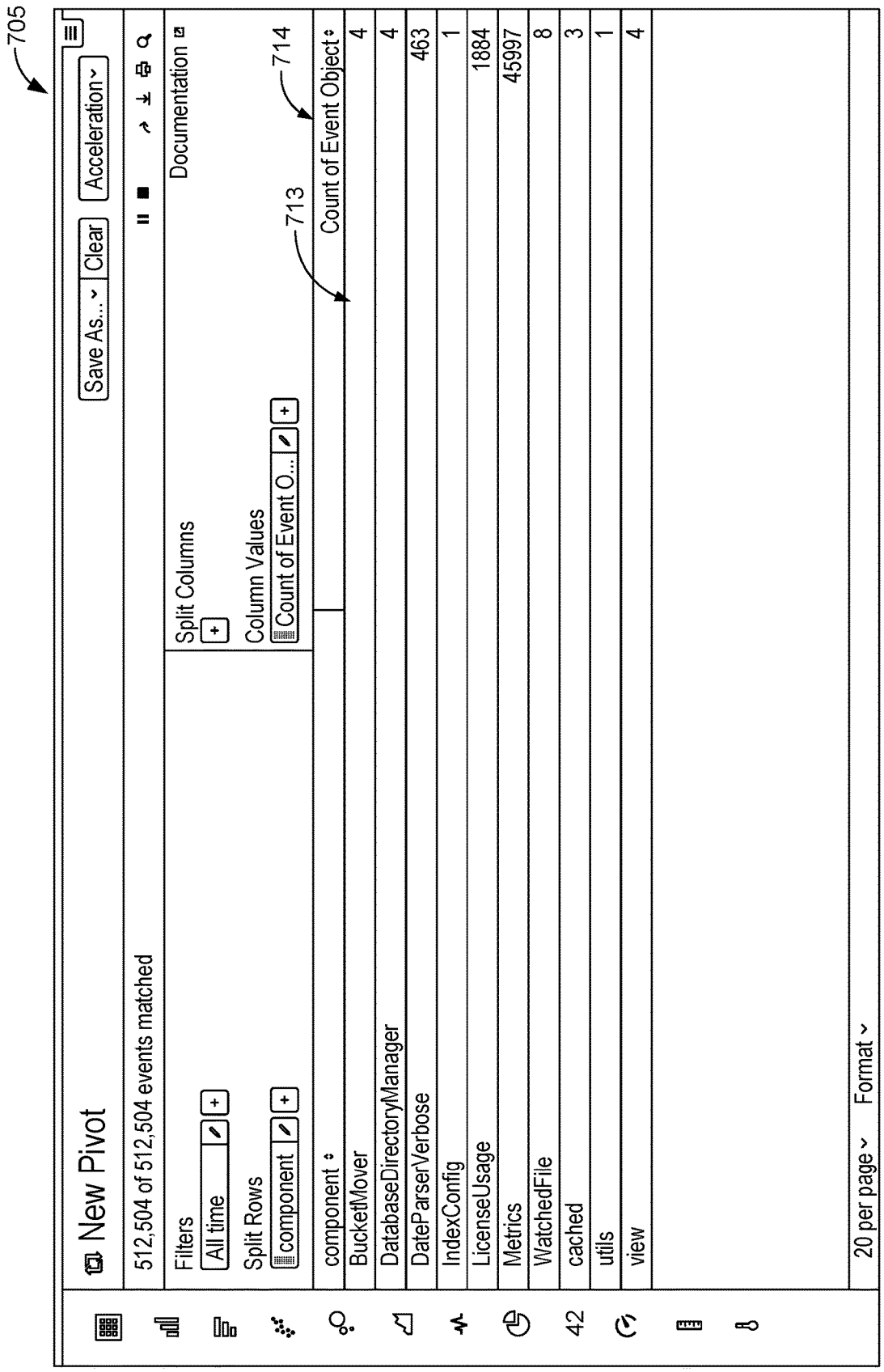

705
New Pivot
Save As...
Clear
Acceleration
512,504 of 512,504 events matched
Documentation
Filters
All time
Split Rows
component
Split Columns
Column Values
Count of Event O...
713
714
component
Count of Event Object
BucketMover 4
DatabaseDirectoryManager 4
DateParserVerbose 463
IndexConfig 1
LicenseUsage 1884
Metrics 45997
WatchedFile 8
cached 3
utils 1
view 4
20 per page
Format

FIG. 7D

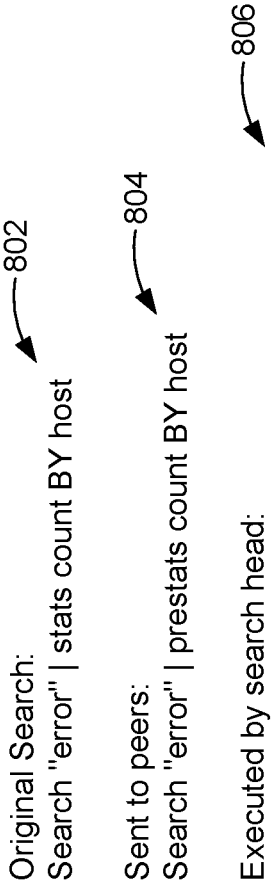

802
Original Search:
Search "error" | stats count BY host
804
Sent to peers:
Search "error" | prestats count BY host
806
Executed by search head:
Aggregate the prestats results received from peers

FIG. 8

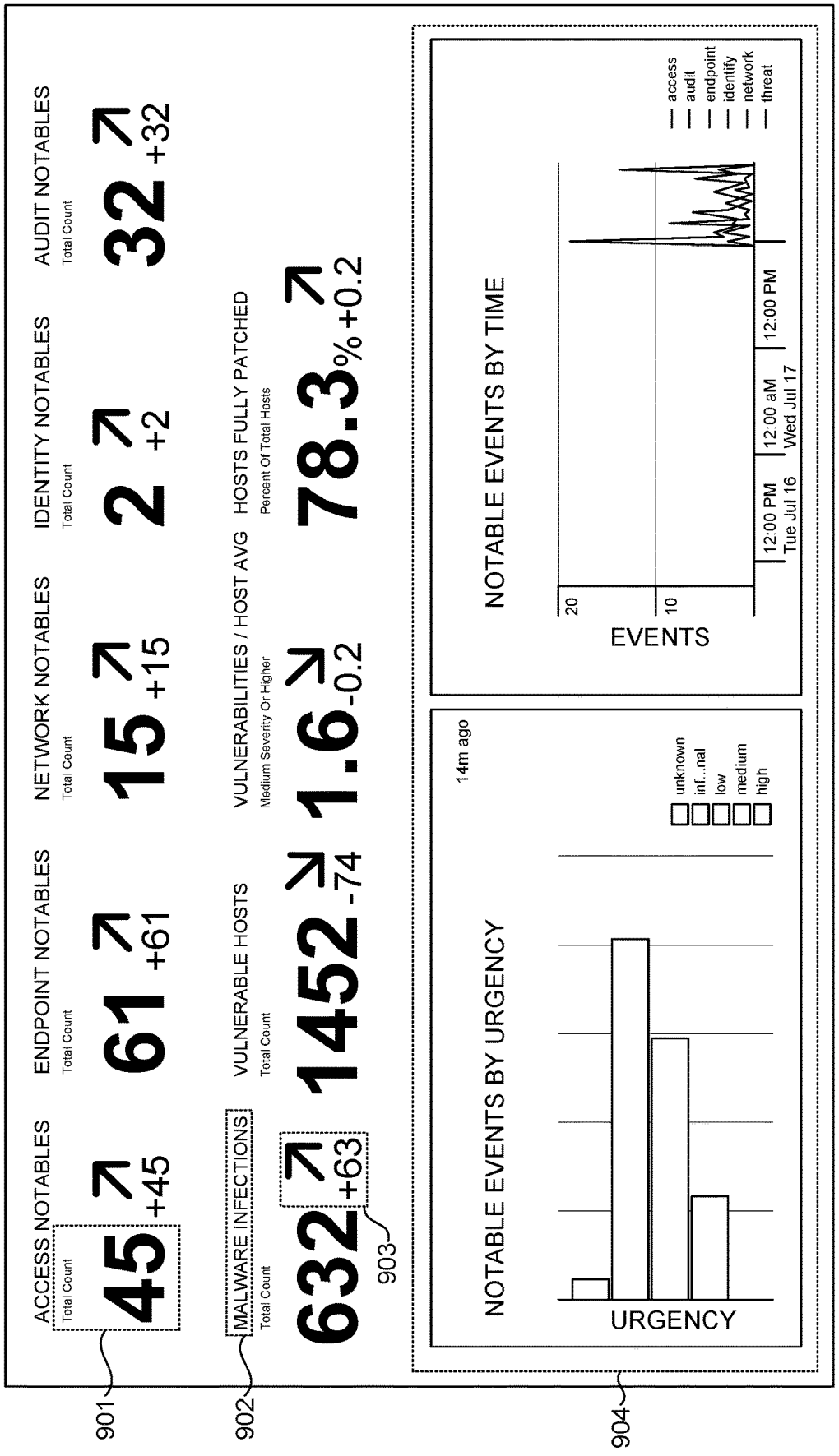

KEY INDICATORS VIEW 900
ACCESS NOTABLES
Total Count
45 +45
ENDPOINT NOTABLES
Total Count
61 +61
NETWORK NOTABLES
Total Count
15 +15
IDENTITY NOTABLES
Total Count
2 +2
AUDIT NOTABLES
Total Count
32 +32
MALWARE INFECTIONS
Total Count
632 +63
VULNERABLE HOSTS
Total Count
1452 -74
VULNERABILITIES / HOST AVG
Medium Severity Or Higher
1.6 -0.2
HOSTS FULLY PATCHED
Percent Of Total Hosts
78.3% +0.2
NOTABLE EVENTS BY URGENCY
14m ago
unknown
inf...nal
low
medium
high
URGENCY
NOTABLE EVENTS BY TIME
access
audit
endpoint
identity
network
threat
20
10
EVENTS
12:00 PM
Tue Jul 16
12:00 aM
Wed Jul 17
12:00 PM
901
902
903
904

FIG. 9A

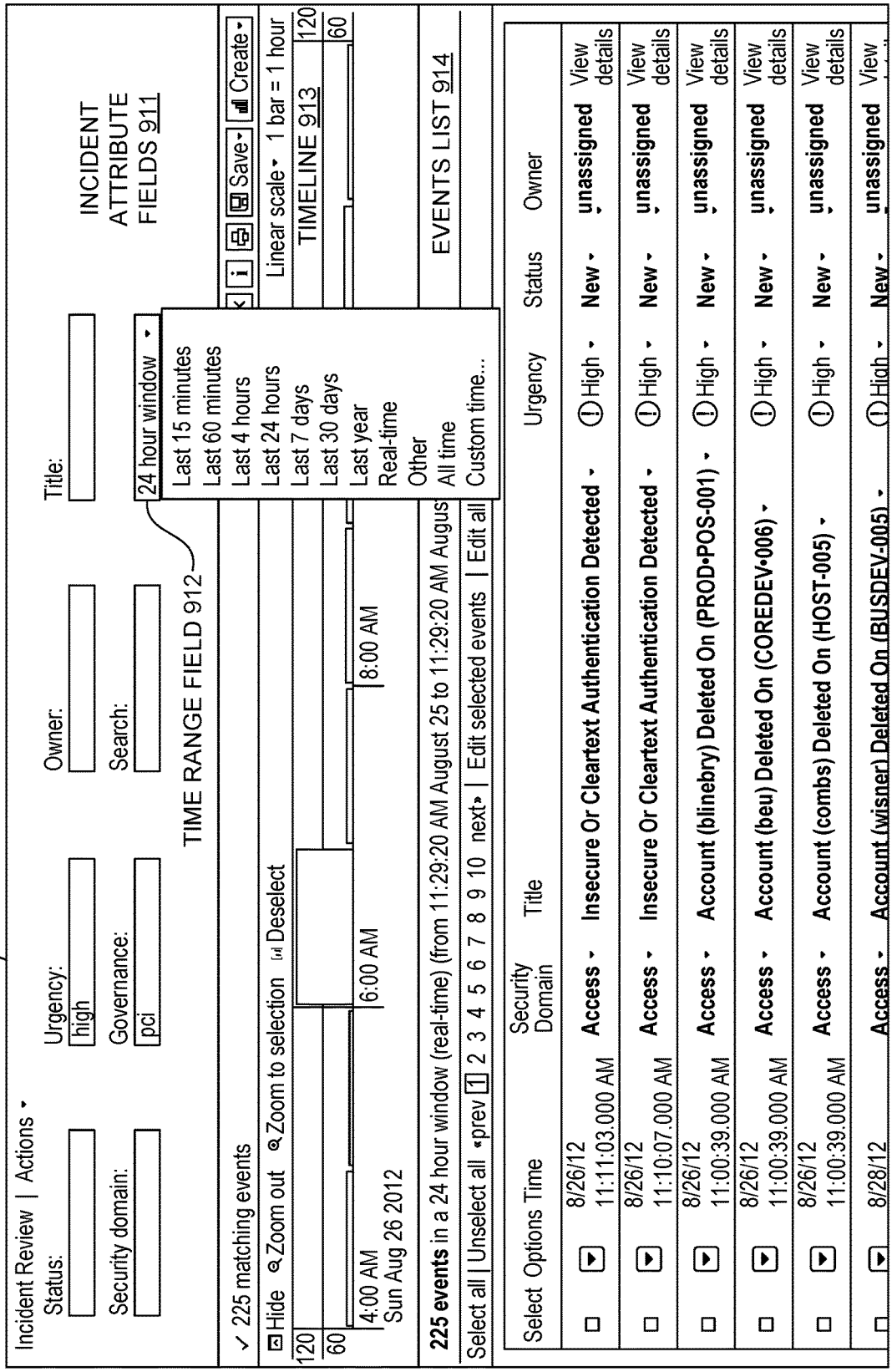

INCIDENT REVIEW DASHBOARD 910
Incident Review | Actions
Status:
Urgency: high
Owner:
Title:
Security domain:
Governance: pci
Search:
INCIDENT ATTRIBUTE FIELDS 911
TIME RANGE FIELD 912
24 hour window
Last 15 minutes
Last 60 minutes
Last 4 hours
Last 24 hours
Last 7 days
Last 30 days
Last year
Real-time
Other
All time
Custom time...
Save
Create
225 matching events
Hide
Zoom out
Zoom to selection
Deselect
Linear scale
1 bar = 1 hour
TIMELINE 913
120
60
4:00 AM
Sun Aug 26 2012
6:00 AM
8:00 AM
EVENTS LIST 914
225 events in a 24 hour window (real-time) (from 11:29:20 AM August 25 to 11:29:20 AM Augus
Select all | Unselect all «prev 1 2 3 4 5 6 7 8 9 10 next» | Edit selected events | Edit all
Select Options Time
Security Domain
Title
Urgency
Status
Owner
8/26/12 11:11:03.000 AM Access Insecure Or Cleartext Authentication Detected High New unassigned View details
8/26/12 11:10:07.000 AM Access Insecure Or Cleartext Authentication Detected High New unassigned View details
8/26/12 11:00:39.000 AM Access Account (blinebry) Deleted On (PROD-POS-001) High New unassigned View details
8/26/12 11:00:39.000 AM Access Account (beu) Deleted On (COREDEV-006) High New unassigned View details
8/26/12 11:00:39.000 AM Access Account (combs) Deleted On (HOST-005) High New unassigned View details
8/28/12 Access Account (wisner) Deleted On (BUSDEV-005) High New unassigned View

FIG. 9B

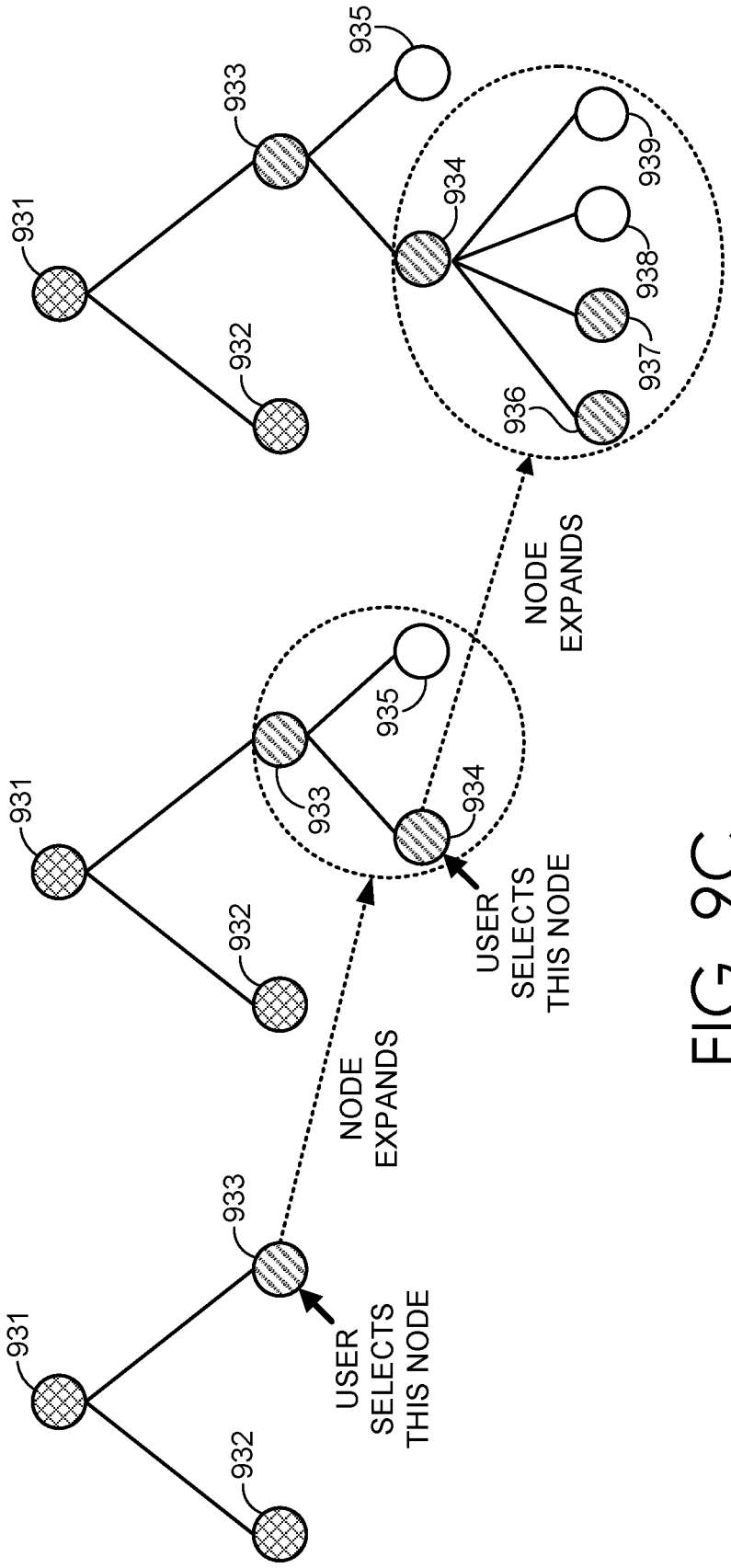

931
932
933
935
934
936
937
938
939
USER SELECTS THIS NODE
NODE EXPANDS
USER SELECTS THIS NODE
NODE EXPANDS

FIG. 9C

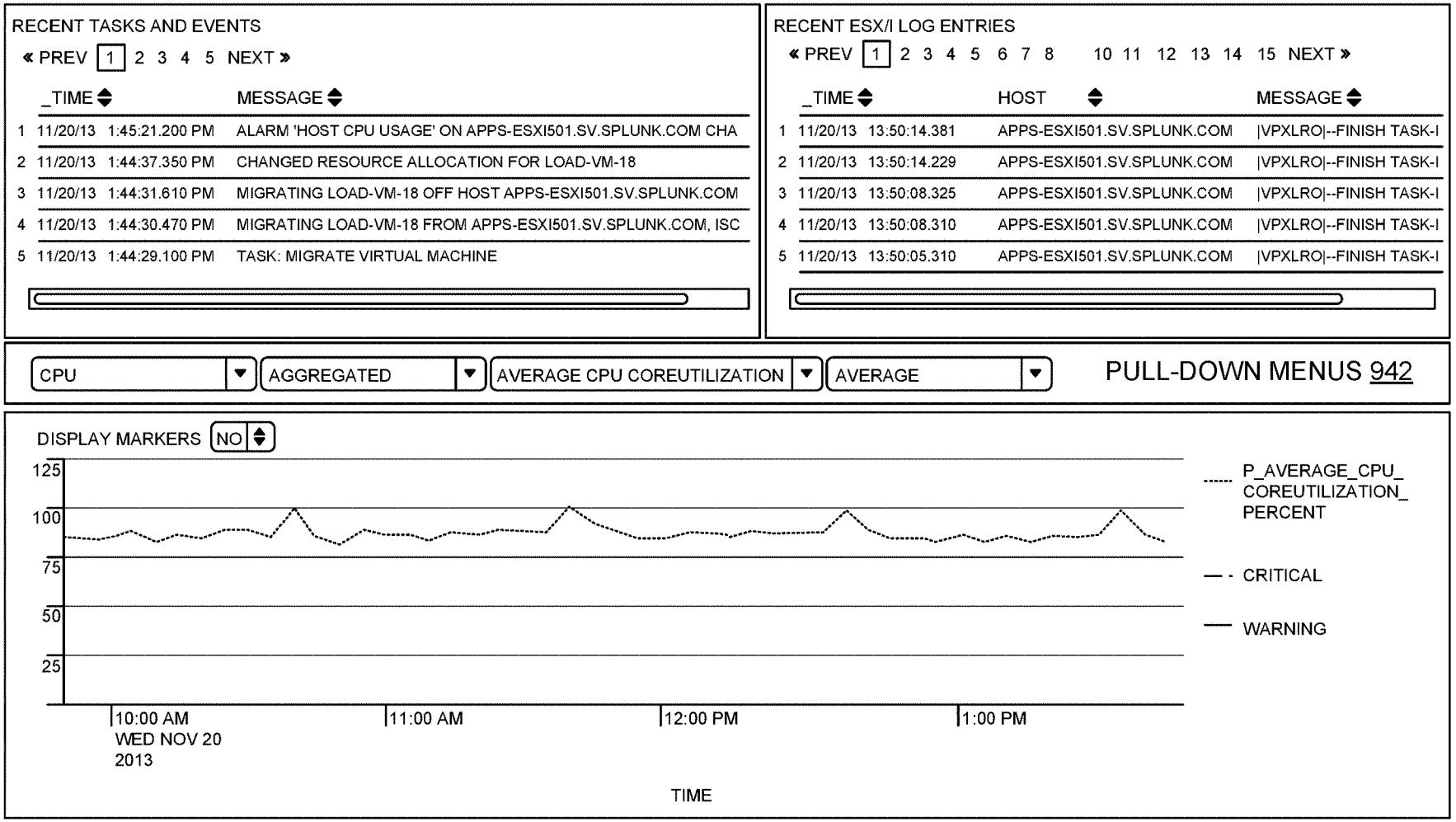

RECENT TASKS AND EVENTS
« PREV 1 2 3 4 5 NEXT »
_TIME
MESSAGE
1 11/20/13 1:45:21.200 PM ALARM 'HOST CPU USAGE' ON APPS-ESXI501.SV.SPLUNK.COM CHA
2 11/20/13 1:44:37.350 PM CHANGED RESOURCE ALLOCATION FOR LOAD-VM-18
3 11/20/13 1:44:31.610 PM MIGRATING LOAD-VM-18 OFF HOST APPS-ESXI501.SV.SPLUNK.COM
4 11/20/13 1:44:30.470 PM MIGRATING LOAD-VM-18 FROM APPS-ESXI501.SV.SPLUNK.COM, ISC
5 11/20/13 1:44:29.100 PM TASK: MIGRATE VIRTUAL MACHINE
RECENT ESX/I LOG ENTRIES
« PREV 1 2 3 4 5 6 7 8 10 11 12 13 14 15 NEXT »
_TIME
HOST
MESSAGE
1 11/20/13 13:50:14.381 APPS-ESXI501.SV.SPLUNK.COM |VPXLRO|--FINISH TASK-I
2 11/20/13 13:50:14.229 APPS-ESXI501.SV.SPLUNK.COM |VPXLRO|--FINISH TASK-I
3 11/20/13 13:50:08.325 APPS-ESXI501.SV.SPLUNK.COM |VPXLRO|--FINISH TASK-I
4 11/20/13 13:50:08.310 APPS-ESXI501.SV.SPLUNK.COM |VPXLRO|--FINISH TASK-I
5 11/20/13 13:50:05.310 APPS-ESXI501.SV.SPLUNK.COM |VPXLRO|--FINISH TASK-I
CPU
AGGREGATED
AVERAGE CPU COREUTILIZATION
AVERAGE
PULL-DOWN MENUS 942
DISPLAY MARKERS NO
125
100
75
50
25
10:00 AM WED NOV 20 2013
11:00 AM
12:00 PM
1:00 PM
TIME
P_AVERAGE_CPU_COREUTILIZATION_PERCENT
CRITICAL
WARNING

FIG. 9D

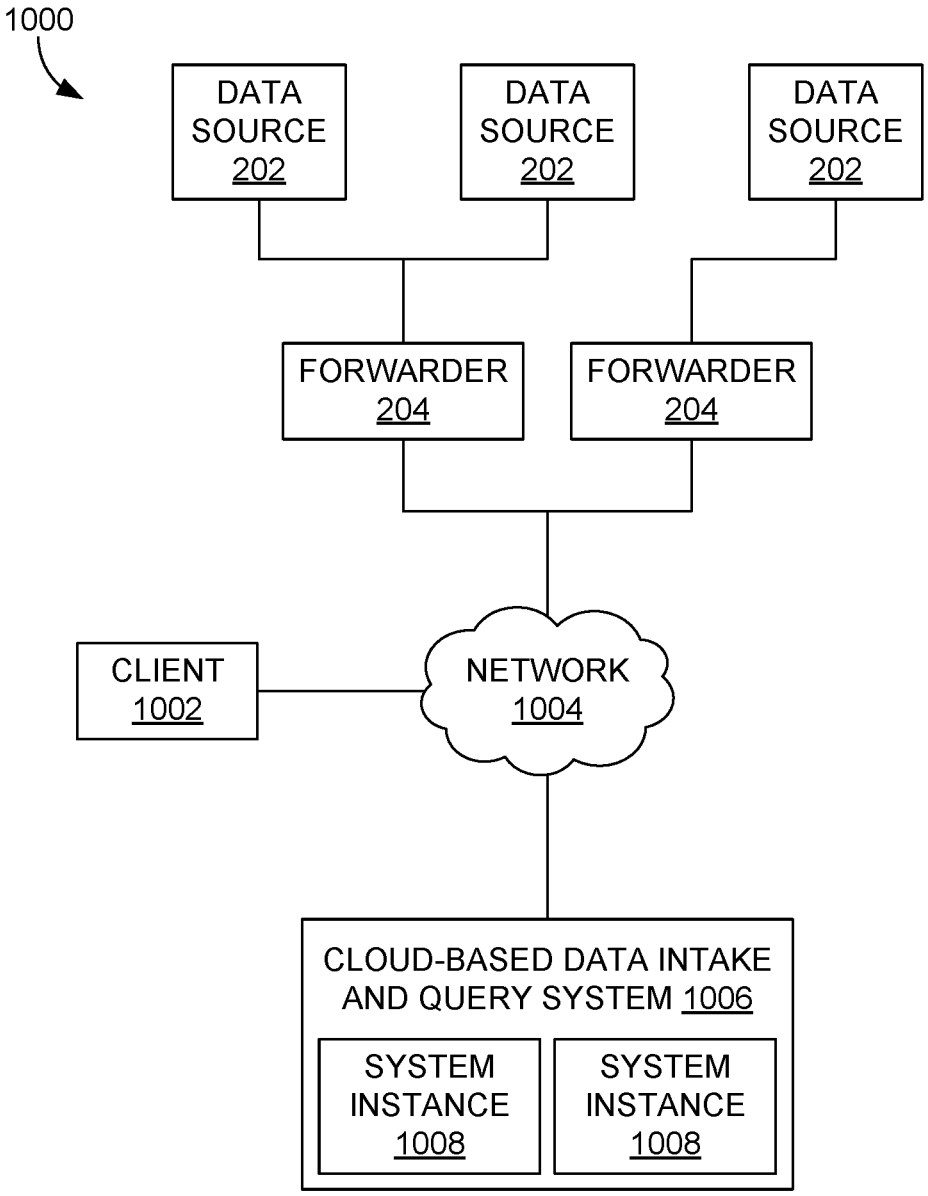

1000
DATA SOURCE 202
DATA SOURCE 202
DATA SOURCE 202
FORWARDER 204
FORWARDER 204
CLIENT 1002
NETWORK 1004
CLOUD-BASED DATA INTAKE AND QUERY SYSTEM 1006
SYSTEM INSTANCE 1008
SYSTEM INSTANCE 1008

FIG. 10

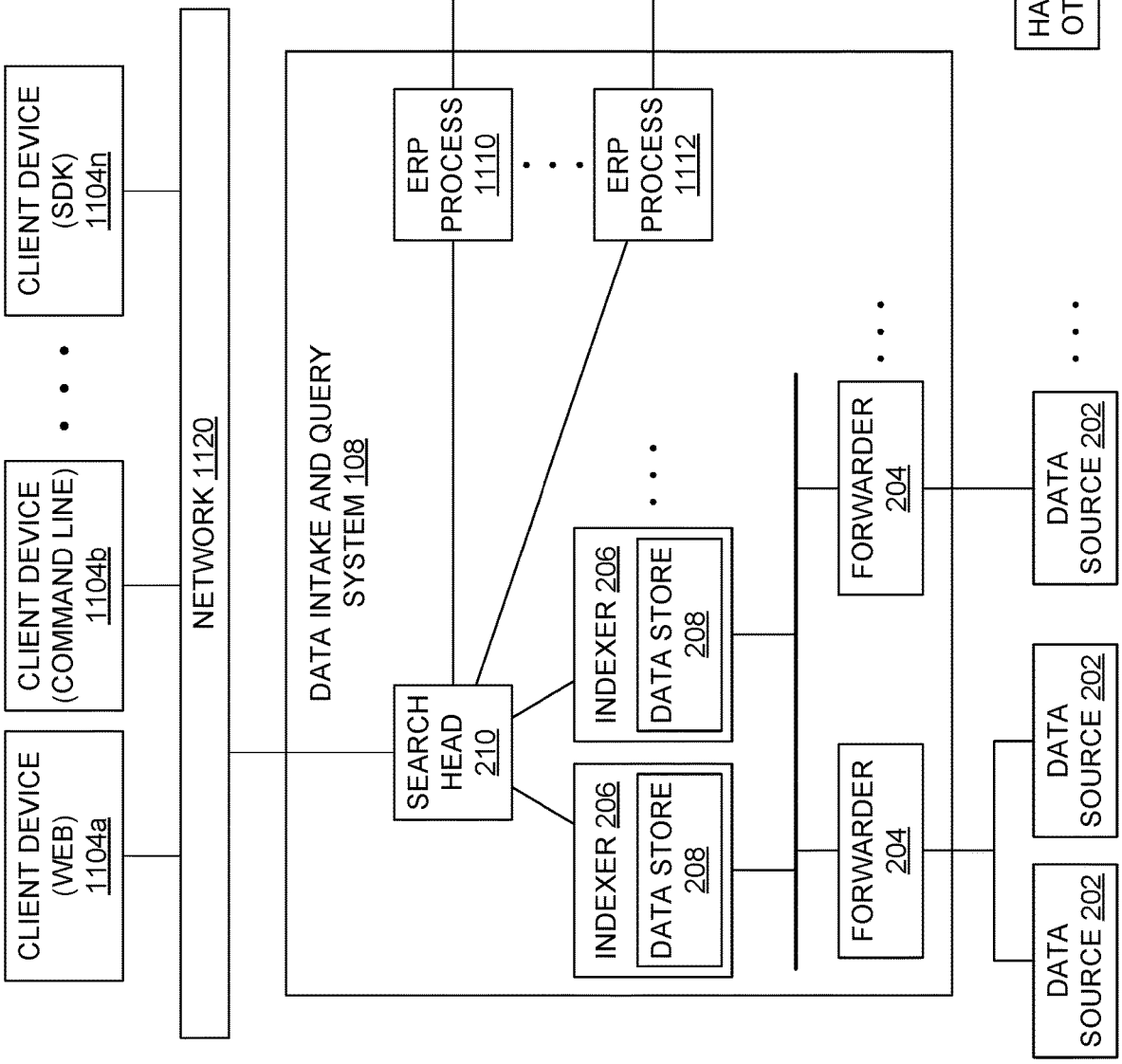

CLIENT DEVICE (WEB) 1104a
CLIENT DEVICE (COMMAND LINE) 1104b
CLIENT DEVICE (SDK) 1104n
NETWORK 1120
DATA INTAKE AND QUERY SYSTEM 108
SEARCH HEAD 210
ERP PROCESS 1110
ERP PROCESS 1112
INDEXER 206
DATA STORE 208
INDEXER 206
DATA STORE 208
FORWARDER 204
FORWARDER 204
DATA SOURCE 202
DATA SOURCE 202
DATA SOURCE 202
HADOOP OR OTHER 1114
RDBMS 1116

FIG. 11

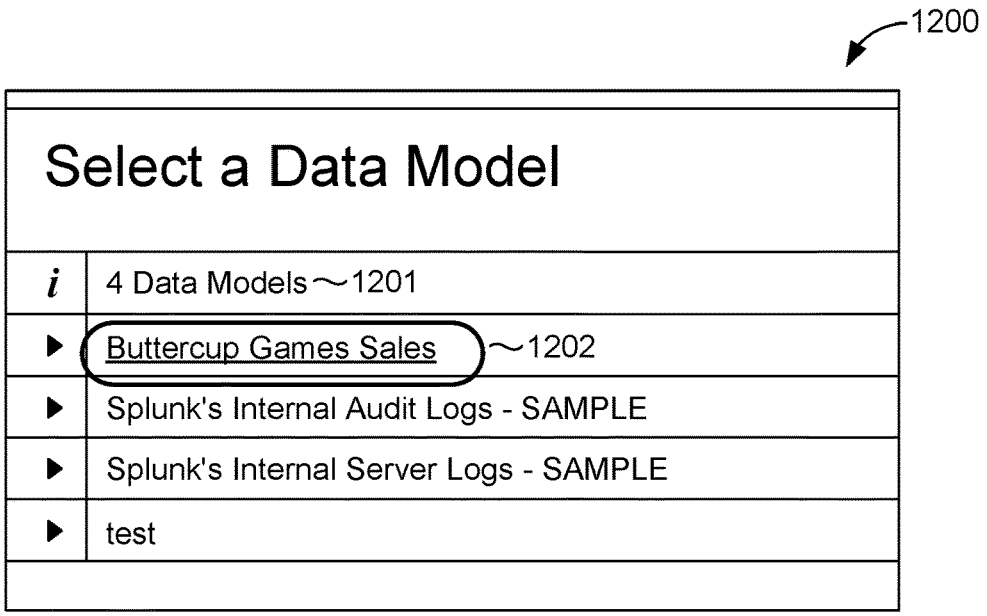

1200
Select a Data Model
i
4 Data Models
1201
Buttercup Games Sales
1202
Splunk's Internal Audit Logs - SAMPLE
Splunk's Internal Server Logs - SAMPLE
test

FIG. 12

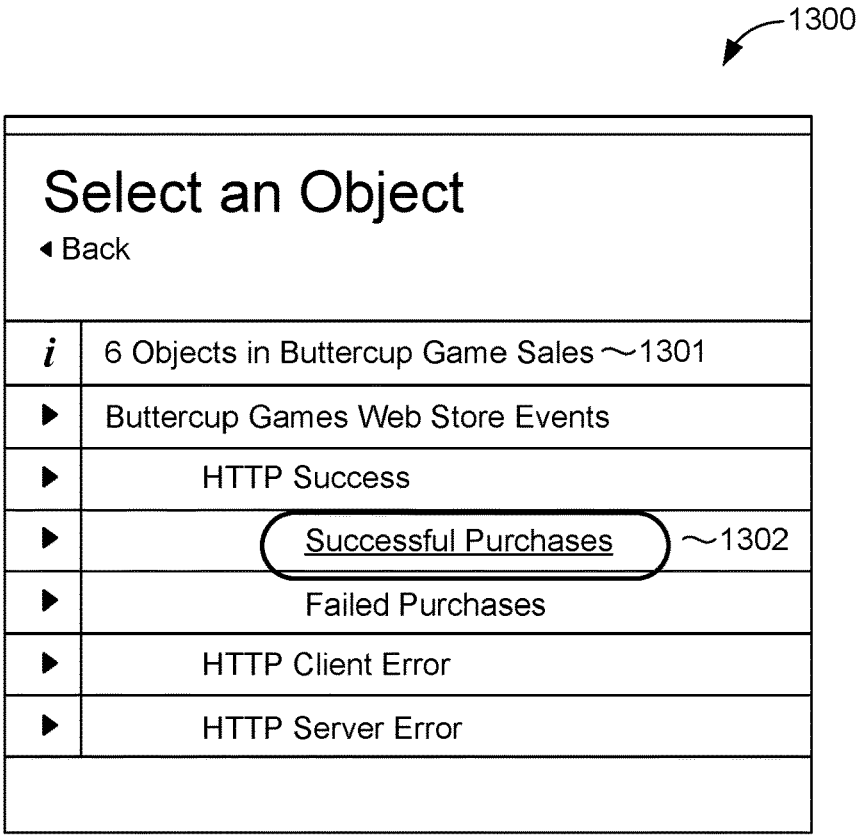

1300
Select an Object
Back
i
6 Objects in Buttercup Game Sales
1301
Buttercup Games Web Store Events
HTTP Success
Successful Purchases
1302
Failed Purchases
HTTP Client Error
HTTP Server Error

FIG. 13

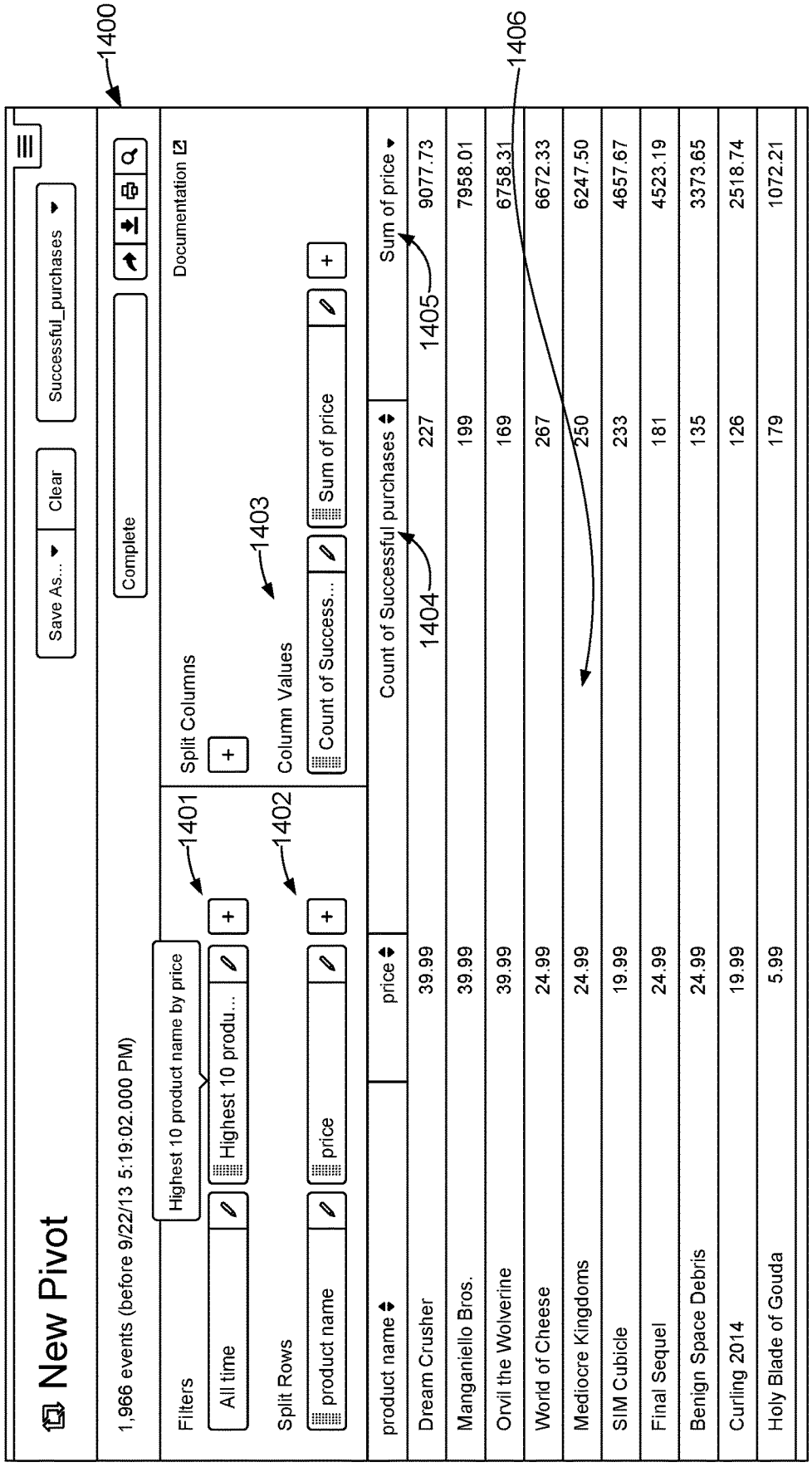

1400
New Pivot
Save As...
Clear
Successful_purchases
1,966 events (before 9/22/13 5:19:02.000 PM)
Complete
Documentation
Filters
All time
Highest 10 product name by price
Highest 10 produ...
1401
Split Rows
product name
price
1402
Split Columns
1403
Column Values
Count of Successful purchases
Sum of price
1404
1405
1406
product name
price
Count of Successful purchases
Sum of price
Dream Crusher 39.99 227 9077.73
Manganiello Bros. 39.99 199 7958.01
Orvil the Wolverine 39.99 169 6758.31
World of Cheese 24.99 267 6672.33
Mediocre Kingdoms 24.99 250 6247.50
SIM Cubicle 19.99 233 4657.67
Final Sequel 24.99 181 4523.19
Benign Space Debris 24.99 135 3373.65
Curling 2014 19.99 126 2518.74
Holy Blade of Gouda 5.99 179 1072.21

FIG. 14

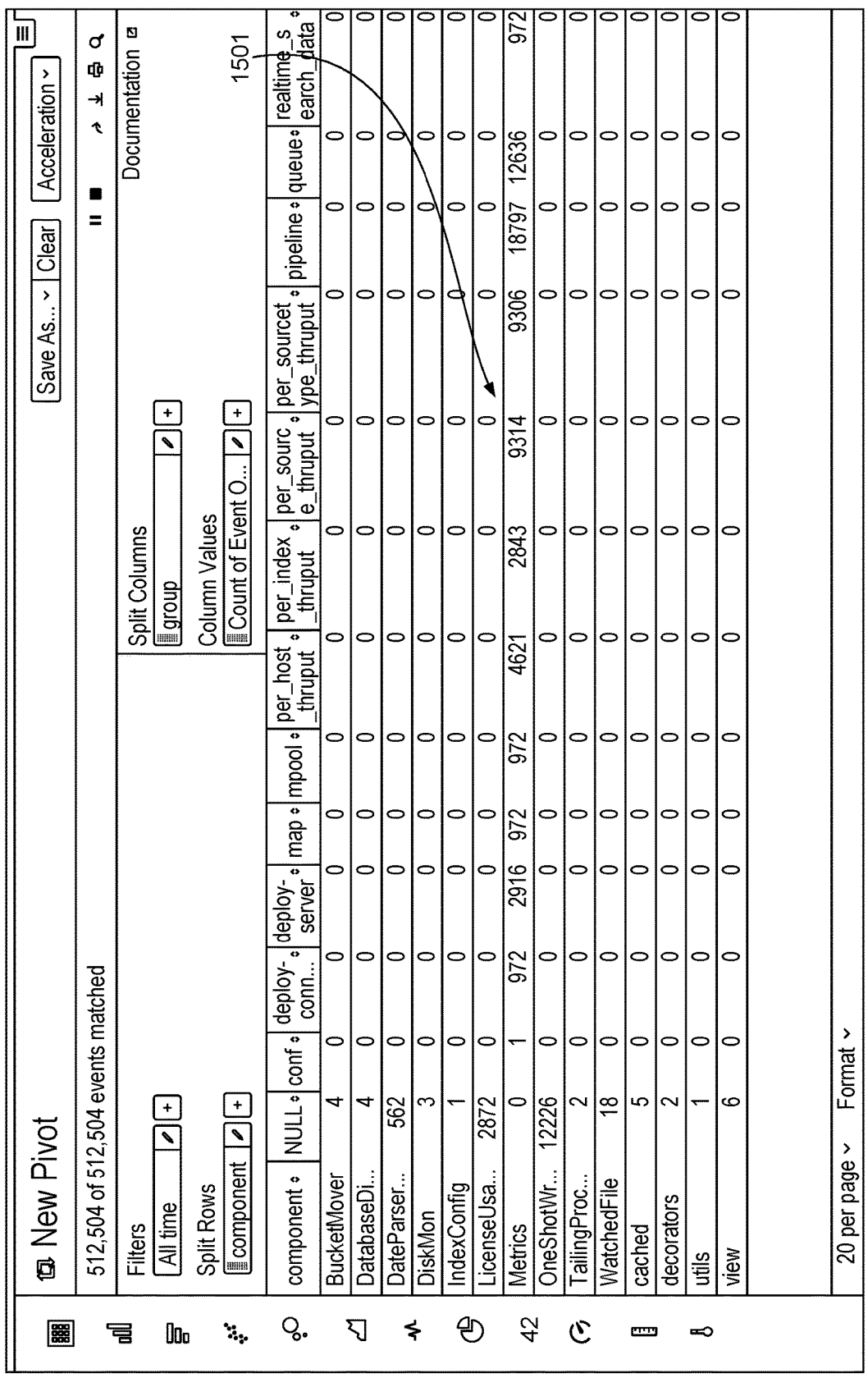

1500
New Pivot
Save As...
Clear
Acceleration
512,504 of 512,504 events matched
Documentation
Filters
All time
Split Rows
component
Split Columns
group
Column Values
Count of Event O...
1501
component
NULL
conf
deploy-conn...
deploy-server
map
mpool
per_host_thruput
per_index_thruput
per_source_thruput
per_sourcetype_thruput
pipeline
queue
realtime_search_data
BucketMover 4 0 0 0 0 0 0 0 0 0 0 0
DatabaseDi... 4 0 0 0 0 0 0 0 0 0 0 0
DateParser... 562 0 0 0 0 0 0 0 0 0 0 0
DiskMon 3 0 0 0 0 0 0 0 0 0 0 0
IndexConfig 1 0 0 0 0 0 0 0 0 0 0 0
LicenseUsa... 2872 0 0 0 0 0 0 0 0 0 0 0
Metrics 0 1 972 2916 972 972 4621 2843 9314 9306 18797 12636 972
OneShotWr... 12226 0 0 0 0 0 0 0 0 0 0 0 0
TailingProc... 2 0 0 0 0 0 0 0 0 0 0 0 0
WatchedFile 18 0 0 0 0 0 0 0 0 0 0 0 0
cached 5 0 0 0 0 0 0 0 0 0 0 0 0
decorators 2 0 0 0 0 0 0 0 0 0 0 0 0
utils 1 0 0 0 0 0 0 0 0 0 0 0 0
view 6 0 0 0 0 0 0 0 0 0 0 0 0
20 per page
Format

FIG. 15

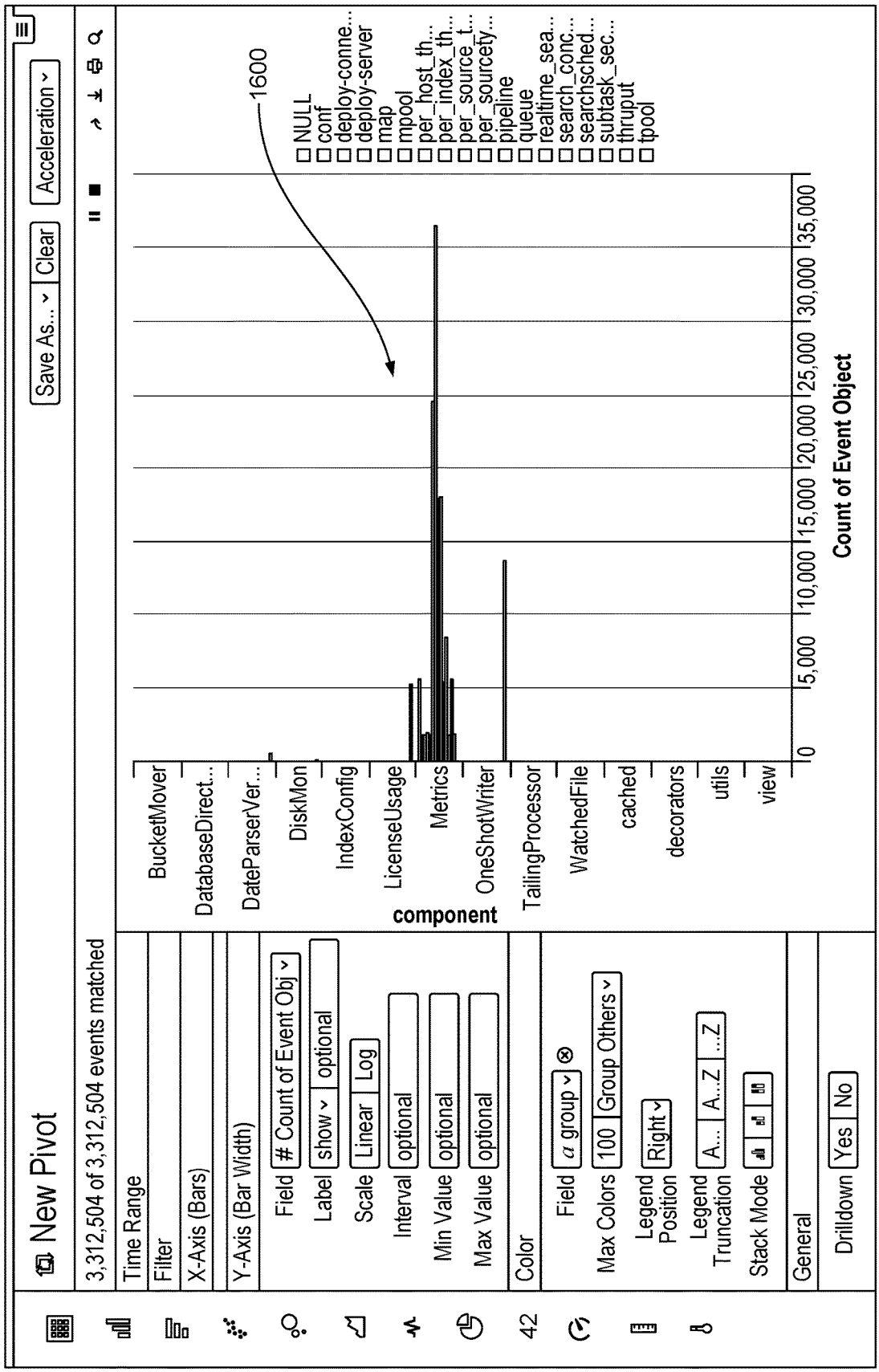

New Pivot
Save As...
Clear
Acceleration
3,312,504 of 3,312,504 events matched
Time Range
Filter
X-Axis (Bars)
Y-Axis (Bar Width)
Field # Count of Event Obj
Label show optional
Scale Linear Log
Interval optional
Min Value optional
Max Value optional
Color
Field group
Max Colors 100 Group Others
Legend Position Right
Legend Truncation A... A...Z ...Z
Stack Mode
General
Drilldown Yes No
BucketMover
DatabaseDirect...
DateParserVer...
DiskMon
IndexConfig
LicenseUsage
Metrics
OneShotWriter
TailingProcessor
WatchedFile
cached
decorators
utils
view
component
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000
Count of Event Object
1600
NULL
conf
deploy-conne...
deploy-server
map
mpool
per_host_th...
per_index_th...
per_source_t...
per_sourcety...
pipeline
queue
realtime_sea...
search_conc...
searchsched...
subtask_sec...
thruput
tpool

FIG. 16

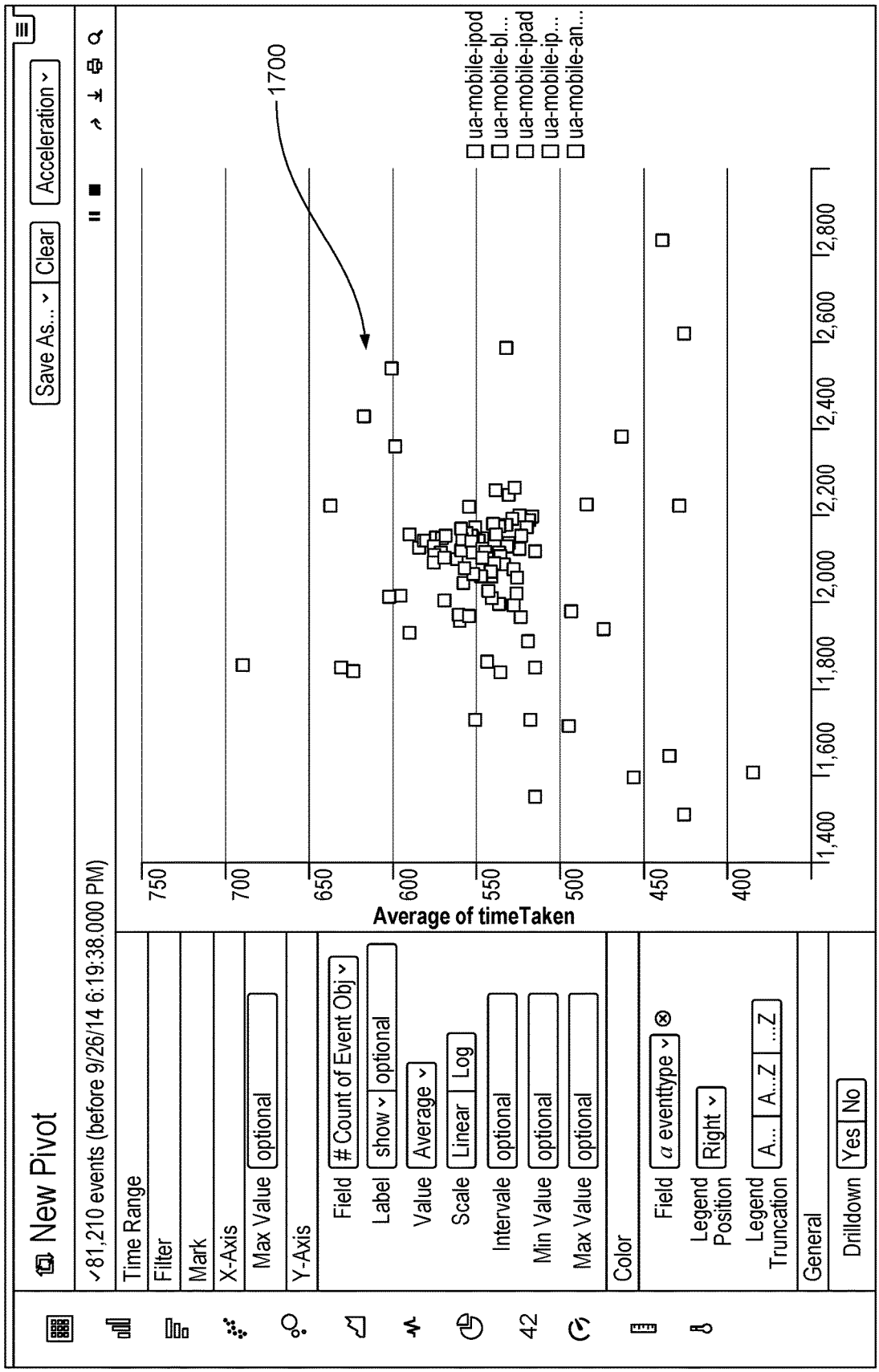
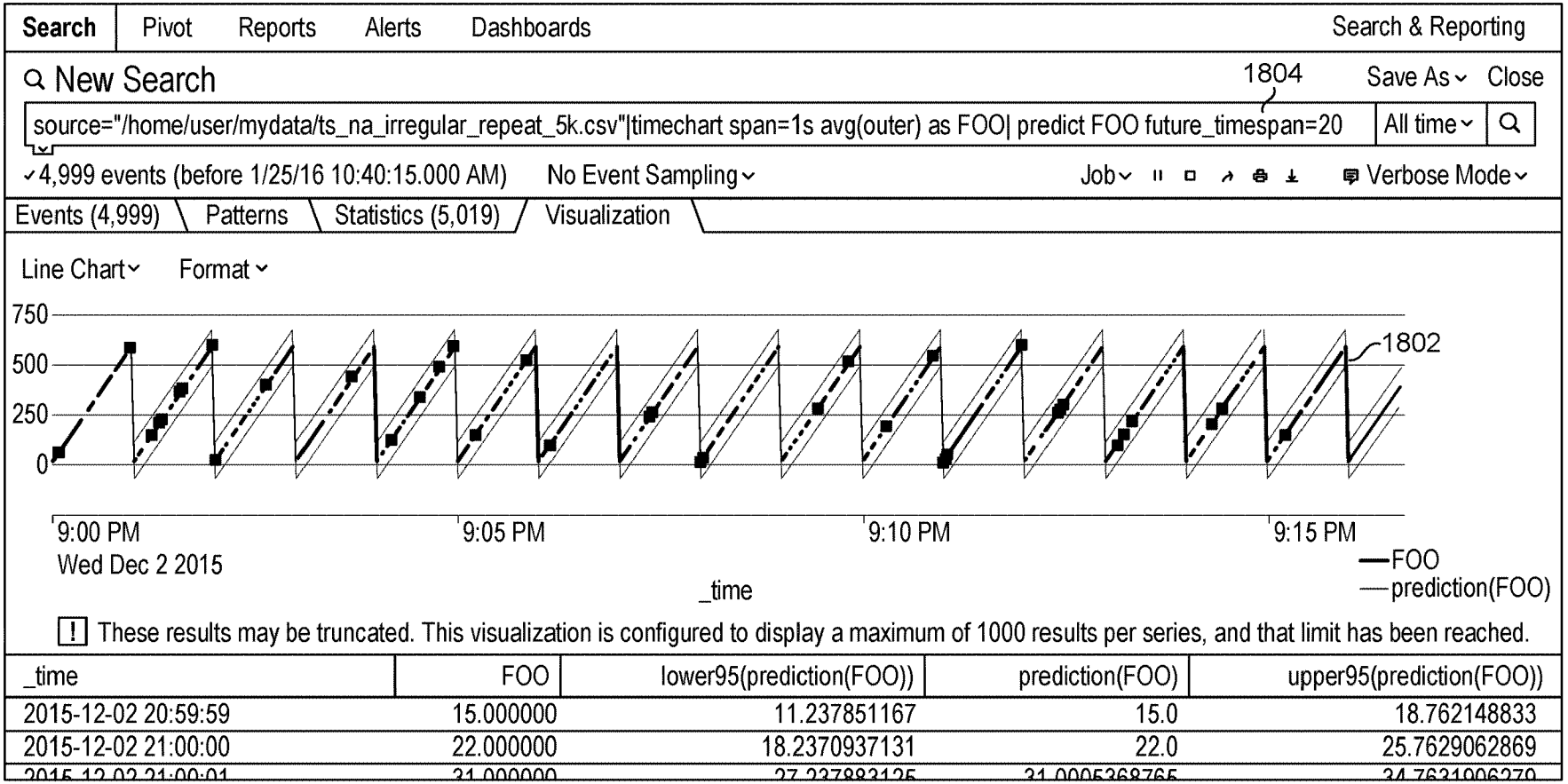

New Pivot
Save As...
Clear
Acceleration
81,210 events (before 9/26/14 6:19:38.000 PM)
Time Range
Filter
Mark
X-Axis
Max Value optional
Y-Axis
Field # Count of Event Obj
Label show optional
Value Average
Scale Linear Log
Interval optional
Min Value optional
Max Value optional
Color
Field a eventtype
Legend Position Right
Legend Truncation A... A...Z ...Z
General
Drilldown Yes No
1700
Average of timeTaken
750
700
650
600
550
500
450
400
1,400
1,600
1,800
2,000
2,200
2,400
2,600
2,800
ua-mobile-ipod
ua-mobile-bl...
ua-mobile-ipad
ua-mobile-ip...
ua-mobile-an...

FIG. 17



_time	FOO	lower95(prediction(FOO))	prediction(FOO)	upper95(prediction(FOO))
2015-12-02 20:59:59	15.000000	11.237851167	15.0	18.762148833
2015-12-02 21:00:00	22.000000	18.2370937131	22.0	25.7629062869
2015-12-02 21:00:01	31.000000	27.237883125	31.0005368765	34.7631006270

FIG. 18A

Search | Pivot | Reports | Alerts | Dashboards | Search & Reporting

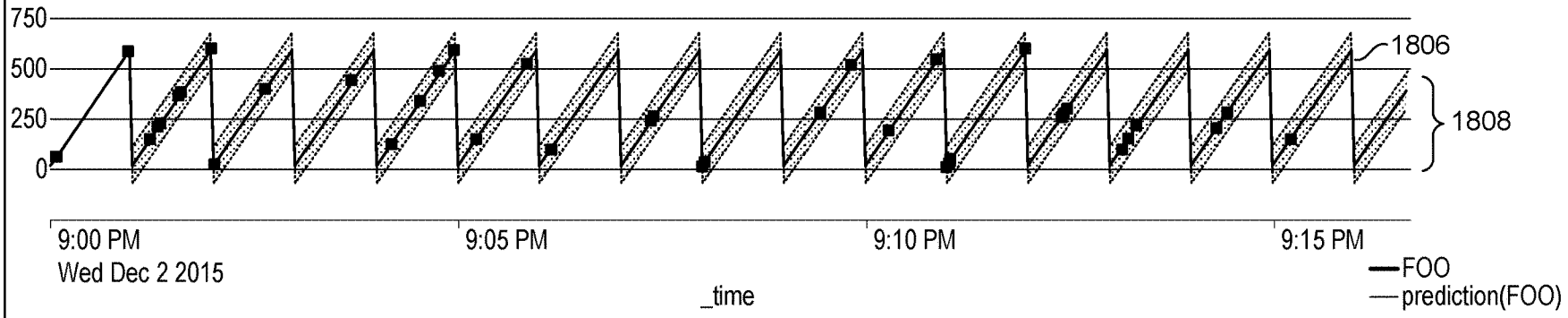
New Search | Save As | Close

source="/home/user/mydata/ts_na_irregular_repeat_5k.csv"|timechart span=1s avg(outer) as FOO| predict FOO future_timespan=20 | All time

✓4,999 events (before 1/25/16 10:40:15.000 AM) | No Event Sampling | Job | Verbose Mode

Events (4,999) | Patterns | Statistics (5,019) | Visualization

Line Chart | Format



! These results may be truncated. This visualization is configured to display a maximum of 1000 results per series, and that limit has been reached.

_time	FOO	lower95(prediction(FOO))	prediction(FOO)	upper95(prediction(FOO))
2015-12-02 20:59:59	15.000000	11.237851167	15.0	18.762148833
2015-12-02 21:00:00	22.000000	18.2370937131	22.0	25.7629062869
2015-12-02 21:00:01	31.000000	27.237883125	31.0005368765	34.7631006270

FIG. 18B

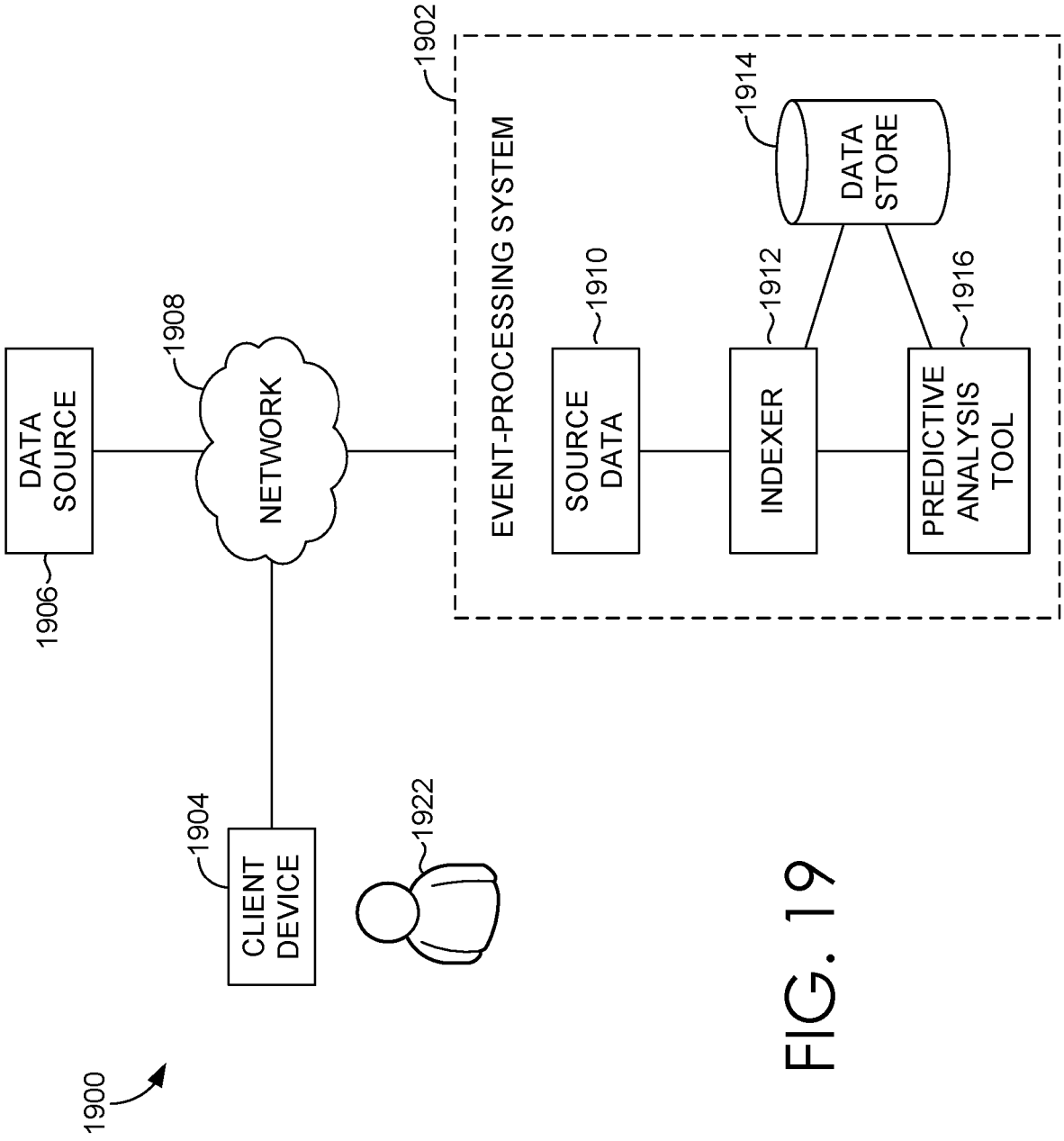
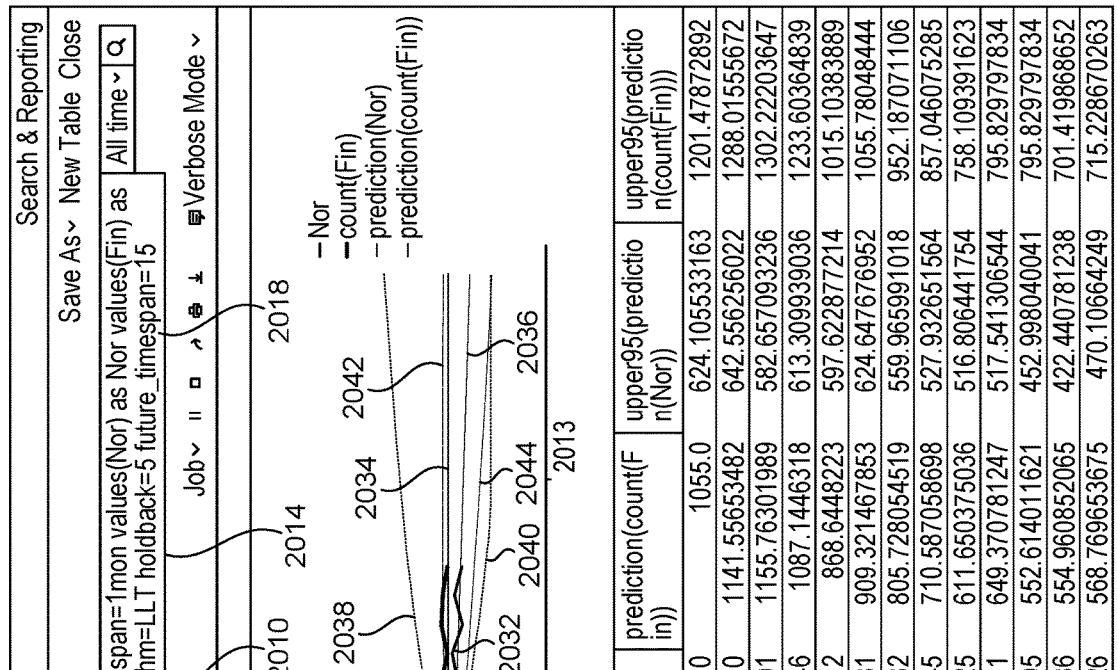

1900
1902
EVENT-PROCESSING SYSTEM
1904
CLIENT DEVICE
1906
DATA SOURCE
1908
NETWORK
1910
SOURCE DATA
1912
INDEXER
1914
DATA STORE
1916
PREDICTIVE ANALYSIS TOOL
1922

FIG. 19



_time	Nor	count(Fin)	lower95(predictio n(Nor))	lower95(predictio n(count(Fin)))	prediction(Nor)	prediction(count(Fin))	upper95(predictio n(Nor))	upper95(predictio n(count(Fin)))
2010-01	560.0	1055.0	495.894466837	908.521271083	560.0	1055.0	624.105533163	1201.47872892
2010-02	533.0	1143.0	423.443743978	995.097512931	533.0	1141.55653482	642.556256022	1288.01555672
2010-03	490.0	1156.0	384.385613146	1009.33040033	483.521353191	1155.76301989	582.657093236	1302.22203647
2010-04	511.0	1086.0	429.463539255	940.685615217	521.386739146	1087.1446318	613.309939036	1233.60364839
2010-05	509.0	865.0	420.669629209	722.185805713	509.146253212	868.6448223	597.622877214	1015.10383889
2010-06	539.0	910.0	451.64326391	762.862451266	538.145470431	909.321467853	624.647676952	1055.78048444
2010-07	471.0	804.0	389.498635147	659.269037931	474.732313082	805.728054519	559.965991018	952.187071106
2010-08	442.0	709.0	359.227733337	564.128042111	443.58019245	710.587058698	527.932651564	857.046075285
2010-09	434.0	610.0	349.395971896	465.191358449	433.101206825	611.650375036	516.806441754	758.109391623
2010-10	437.0	650.0	351.121615655	502.91176466	434.3114611	649.370781247	517.541306544	795.829797834
2010-11	362.0	551.0	287.361062969	406.154995034	370.179551505	552.614011621	452.998040041	795.829797834
2010-12	338.0	555.0	257.437768134	408.501835478	339.939274686	554.960852065	422.440781238	701.419868652
2011-01	401.0	569.0	305.627574029	422.310637088	387.86710826	568.769653675	470.10664249	715.228670263

2020

2046

FIG. 20

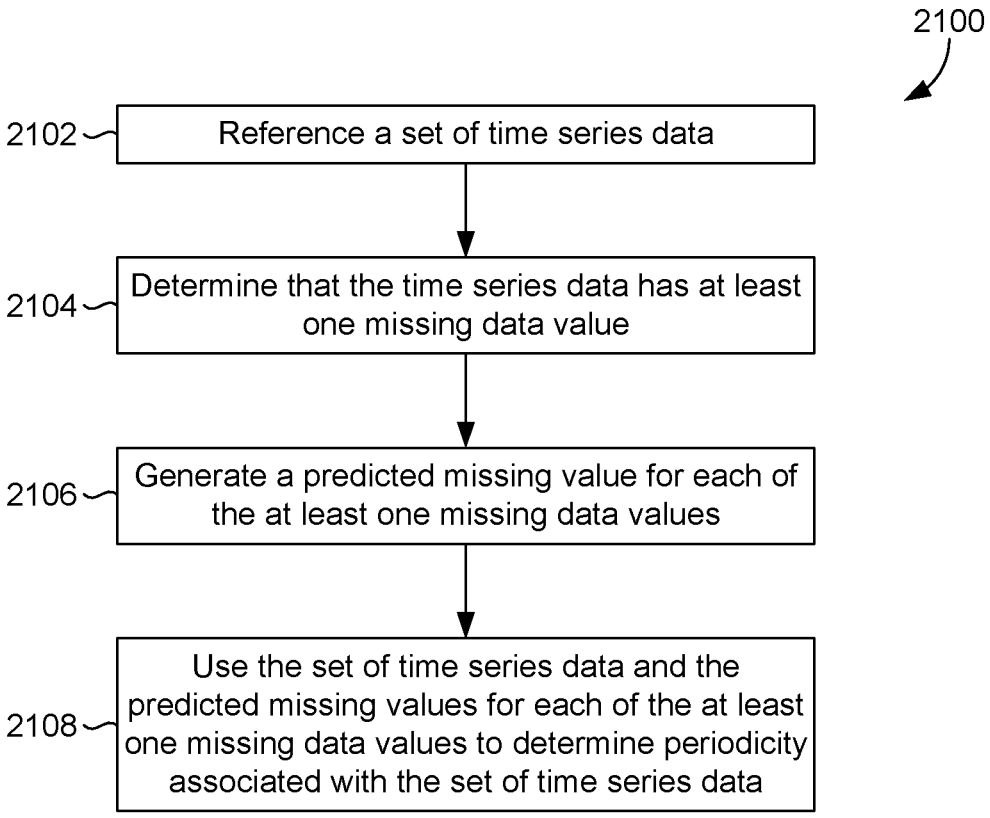

2100
2102
Reference a set of time series data
2104
Determine that the time series data has at least one missing data value
2106
Generate a predicted missing value for each of the at least one missing data values
2108
Use the set of time series data and the predicted missing values for each of the at least one missing data values to determine periodicity associated with the set of time series data

FIG. 21

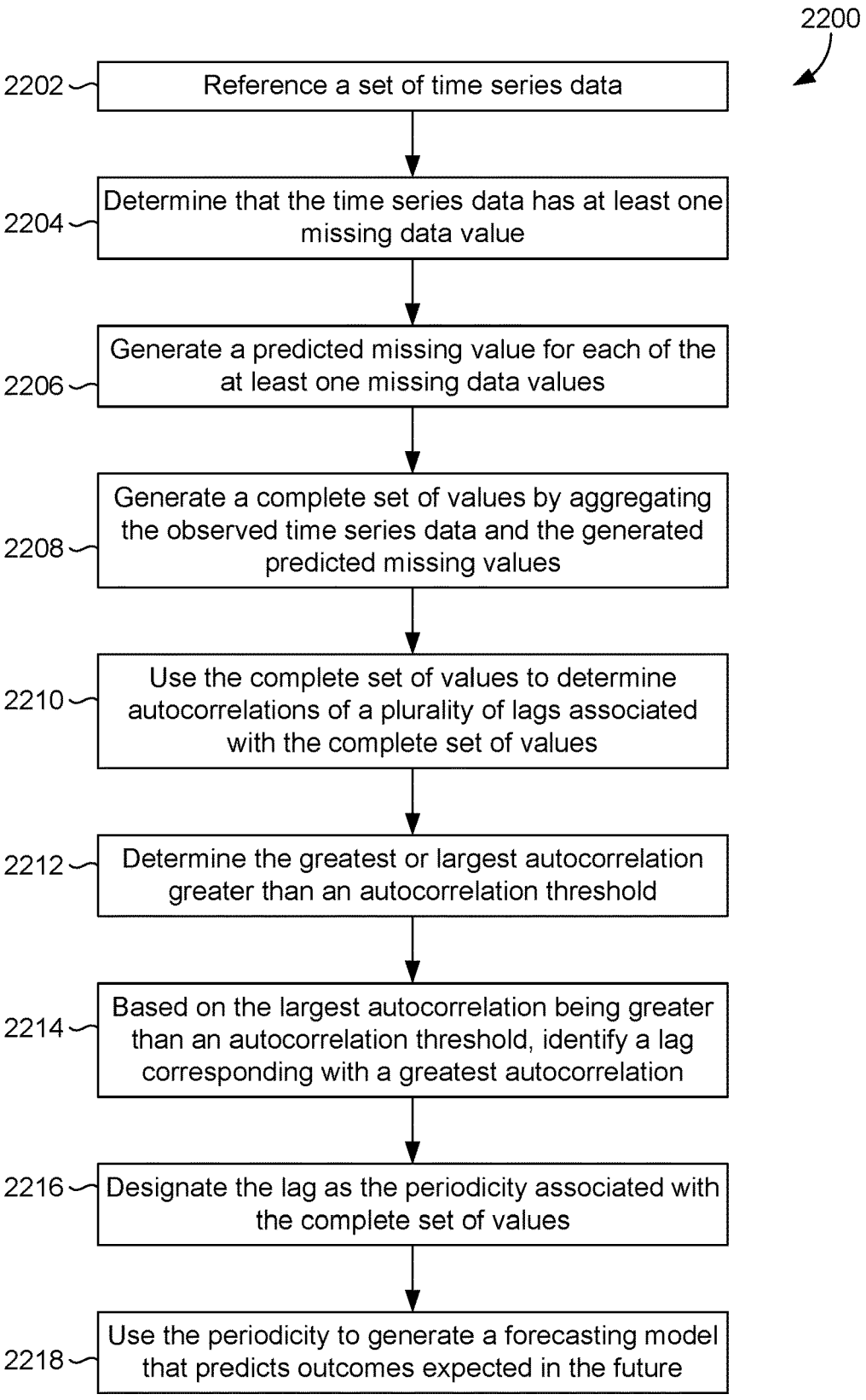

2200
2202
Reference a set of time series data
2204
Determine that the time series data has at least one missing data value
2206
Generate a predicted missing value for each of the at least one missing data values
2208
Generate a complete set of values by aggregating the observed time series data and the generated predicted missing values
2210
Use the complete set of values to determine autocorrelations of a plurality of lags associated with the complete set of values
2212
Determine the greatest or largest autocorrelation greater than an autocorrelation threshold
2214
Based on the largest autocorrelation being greater than an autocorrelation threshold, identify a lag corresponding with a greatest autocorrelation
2216
Designate the lag as the periodicity associated with the complete set of values
2218
Use the periodicity to generate a forecasting model that predicts outcomes expected in the future

FIG. 22

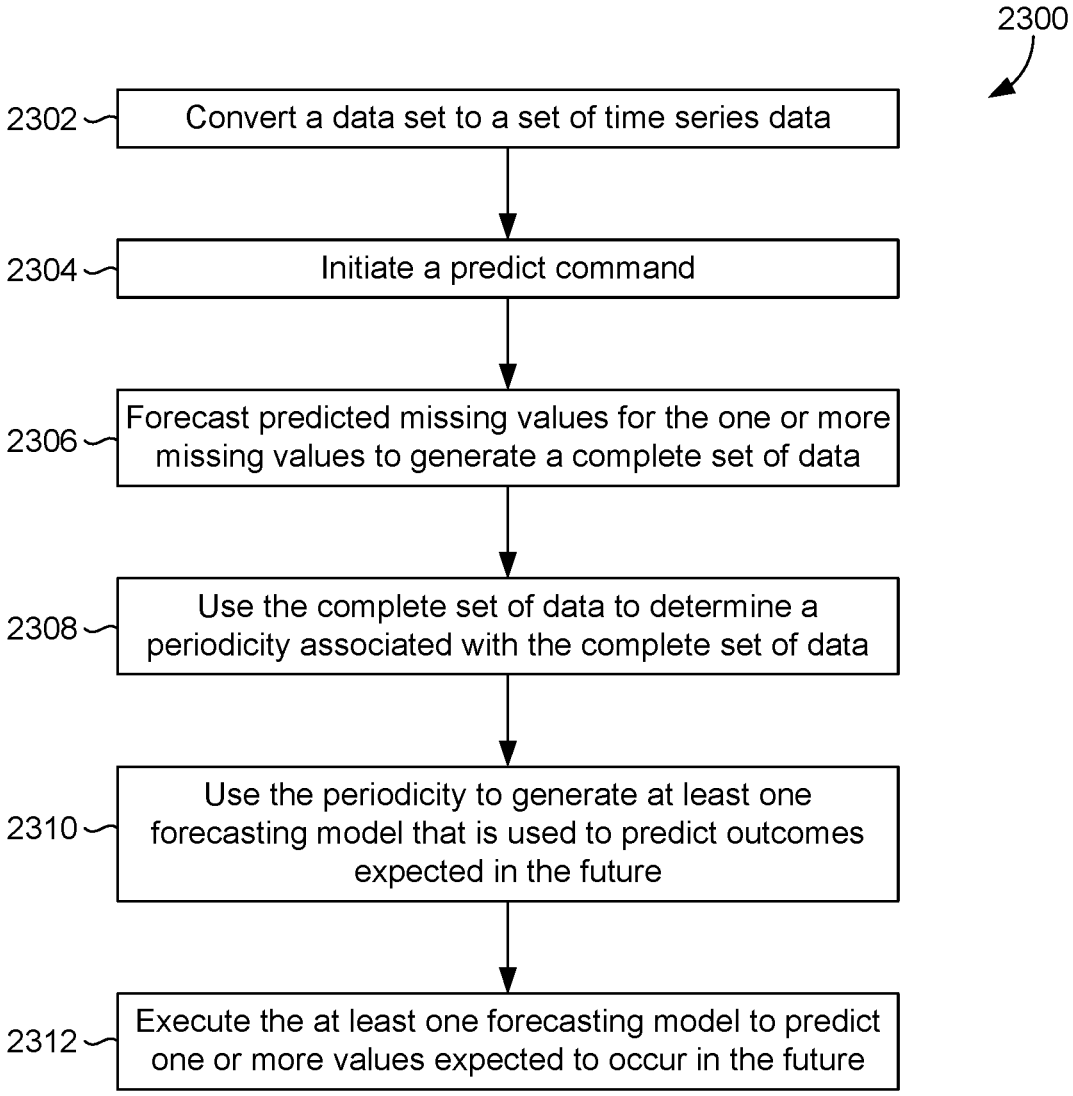

2300
2302
Convert a data set to a set of time series data
2304
Initiate a predict command
2306
Forecast predicted missing values for the one or more missing values to generate a complete set of data
2308
Use the complete set of data to determine a periodicity associated with the complete set of data
2310
Use the periodicity to generate at least one forecasting model that is used to predict outcomes expected in the future
2312
Execute the at least one forecasting model to predict one or more values expected to occur in the future

FIG. 23

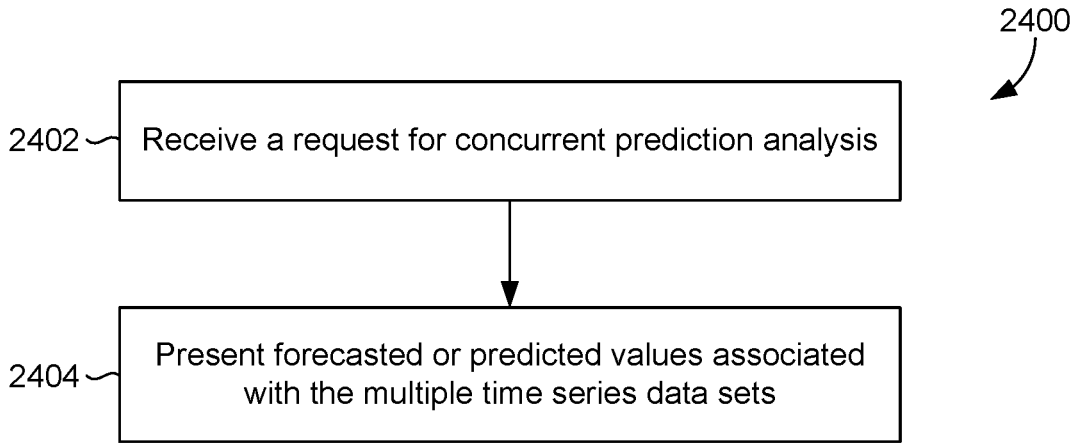

2400
2402
Receive a request for concurrent prediction analysis
2404
Present forecasted or predicted values associated with the multiple time series data sets

FIG. 24

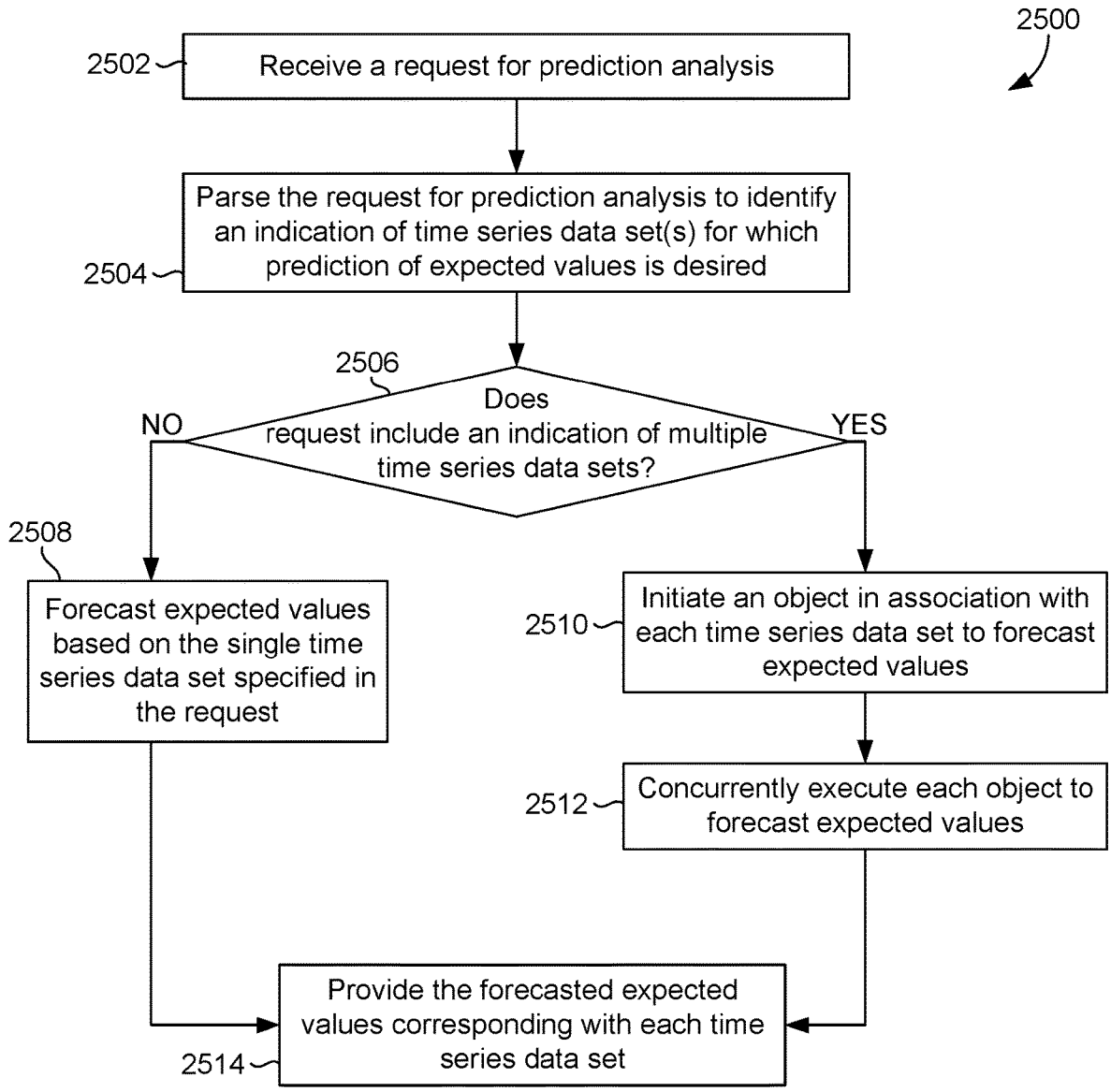

2500
2502
Receive a request for prediction analysis
Parse the request for prediction analysis to identify an indication of time series data set(s) for which prediction of expected values is desired
2504
2506
Does request include an indication of multiple time series data sets?
NO
YES
2508
Forecast expected values based on the single time series data set specified in the request
2510
Initiate an object in association with each time series data set to forecast expected values
2512
Concurrently execute each object to forecast expected values
Provide the forecasted expected values corresponding with each time series data set
2514

FIG. 25

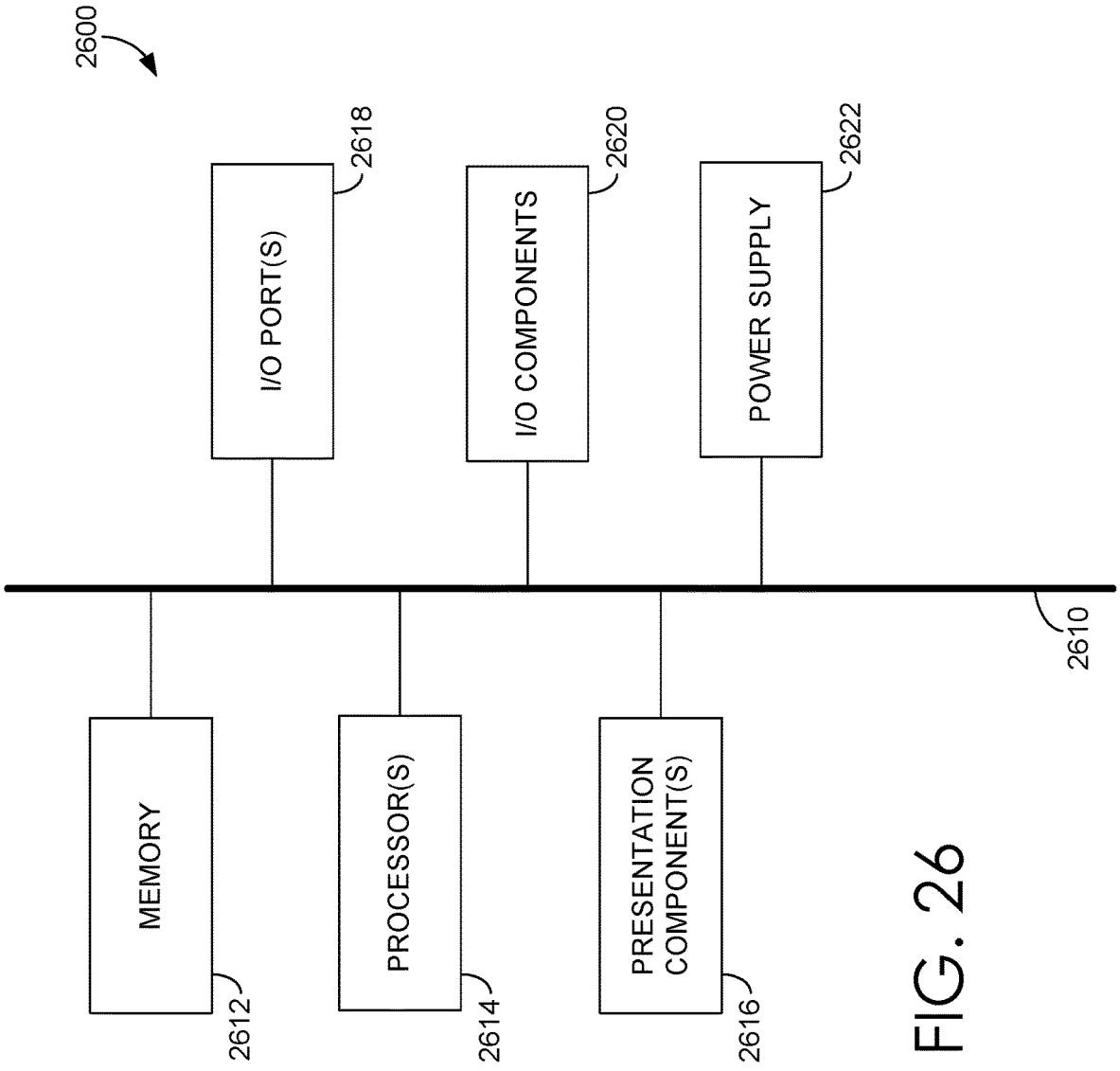

2600
I/O PORT(S)
2618
I/O COMPONENTS
2620
POWER SUPPLY
2622
2610
MEMORY
2612
PROCESSOR(S)
2614
PRESENTATION COMPONENT(S)
2616

FIG. 26

FACILITATING CONCURRENT FORECASTING OF MULTIPLE TIME SERIES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. application Ser. No. 15/143,335, filed Apr. 29, 2016 and titled "CONCURRENTLY FORECASTING MULTIPLE TIME SERIES," which is itself a continuation-in-part of U.S. application Ser. No. 15/010,732, filed Jan. 29, 2016 and titled "ENHANCING TIME SERIES PREDICTION," the contents of each of the foregoing applications are incorporated by reference herein in their entirety.

BACKGROUND

Modern data centers often include thousands of hosts that operate collectively to service requests from even larger numbers of remote clients. During operation, components of these data centers can produce significant volumes of raw, machine-generated data. In many cases, a user wishes to predict or forecast future values from such collected data.

SUMMARY

Embodiments of the present invention are related to facilitating concurrent forecasting associated with multiple time series data sets. In accordance with aspects of the present disclosure, a request to perform a predictive analysis in association with multiple time series data sets is received. Thereafter, the request is parsed to identify each of the time series data sets to use in predictive analyses. For each time series data set, an object is initiated to perform the predictive analysis for the corresponding time series data set. Generally, the predictive analysis predicts expected outcomes based on the corresponding time series data set. Each object is concurrently executed to generate expected outcomes associated with the corresponding time series data set, and the expected outcomes associated with each of the corresponding time series data sets are provided for display.

This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings:

FIG. **1** illustrates a networked computer environment in which an embodiment may be implemented;

FIG. **2** illustrates a block diagram of an example data intake and query system in which an embodiment may be implemented;

FIG. **3** is a flow diagram that illustrates how indexers process, index, and store data received from forwarders in accordance with the disclosed embodiments;

FIG. **4** is a flow diagram that illustrates how a search head and indexers perform a search query in accordance with the disclosed embodiments;

FIG. **5** illustrates a scenario where a common customer ID is found among log data received from three disparate sources in accordance with the disclosed embodiments;

FIG. **6**A illustrates a search screen in accordance with the disclosed embodiments;

FIG. **6**B illustrates a data summary dialog that enables a user to select various data sources in accordance with the disclosed embodiments;

FIGS. **7**A-**7**D illustrate a series of user interface screens for an example data model-driven report generation interface in accordance with the disclosed embodiments;

FIG. **8** illustrates an example search query received from a client and executed by search peers in accordance with the disclosed embodiments;

FIG. **9**A illustrates a key indicators view in accordance with the disclosed embodiments;

FIG. **9**B illustrates an incident review dashboard in accordance with the disclosed embodiments;

FIG. **9**C illustrates a proactive monitoring tree in accordance with the disclosed embodiments;

FIG. **9**D illustrates a user interface screen displaying both log data and performance data in accordance with the disclosed embodiments;

FIG. **10** illustrates a block diagram of an example cloud-based data intake and query system in which an embodiment may be implemented;

FIG. **11** illustrates a block diagram of an example data intake and query system that performs searches across external data systems in accordance with the disclosed embodiments;

FIGS. **12-14** illustrate a series of user interface screens for an example data model-driven report generation interface in accordance with the disclosed embodiments;

FIGS. **15-17** illustrate example visualizations generated by a reporting application in accordance with the disclosed embodiments;

FIG. **18**A illustrates an exemplary user interface showing a portion of a time series data set with indications of values missing therefrom, in accordance with embodiments of the present invention;

FIG. **18**B illustrates an exemplary user interface showing a portion of a time series data set with predicted missing values and predicted expected outcomes, in accordance with embodiments of the present invention;

FIG. **19** depicts a block diagram of an illustrative data processing environment in accordance with various embodiments of the present disclosure;

FIG. **20** illustrates an exemplary user interface showing a concurrent display of forecasted values associated with multiple time series data sets, in accordance with embodiments of the present invention;

FIG. **21** is a flow diagram depicting an illustrative method of computing periodicity utilizing predicted missing values, according to embodiments of the present invention;

FIG. **22** is a flow diagram depicting another method of computing periodicity utilizing predicted missing values, according to embodiments of the present invention;

FIG. **23** is a flow diagram depicting an illustrative method of predicting expected values, in accordance with embodiments of the present invention;

FIG. **24** is a flow diagram depicting an illustrative method of facilitating concurrent predictive analysis for multiple time series data sets, in accordance with embodiments of the present invention;

FIG. **25** is a flow diagram depicting another illustrative method of facilitating concurrent predictive analysis for multiple time series data sets, in accordance with embodiments of the present invention; and

FIG. **26** is a block diagram of an example computing device in which embodiments of the present disclosure may be employed.

DETAILED DESCRIPTION

Embodiments are described herein according to the following outline:

1.0 General Overview

Modern data centers and other computing environments can comprise anywhere from a few host computer systems to thousands of systems configured to process data, service requests from remote clients, and perform numerous other computational tasks. During operation, various components within these computing environments often generate significant volumes of machine-generated data. For example, machine data is generated by various components in the information technology (IT) environments, such as servers, sensors, routers, mobile devices, Internet of Things (IoT) devices, etc. Machine-generated data can include system logs, network packet data, sensor data, application program data, error logs, stack traces, system performance data, etc. In general, machine-generated data can also include performance data, diagnostic information, and many other types of data that can be analyzed to diagnose performance problems, monitor user interactions, and to derive other insights.

A number of tools are available to analyze machine data, that is, machine-generated data. In order to reduce the size of the potentially vast amount of machine data that may be generated, many of these tools typically pre-process the data based on anticipated data-analysis needs. For example, pre-specified data items may be extracted from the machine data and stored in a database to facilitate efficient retrieval and analysis of those data items at search time. However, the rest of the machine data typically is not saved and discarded during pre-processing. As storage capacity becomes progressively cheaper and more plentiful, there are fewer incentives to discard these portions of machine data and many reasons to retain more of the data.

This plentiful storage capacity is presently making it feasible to store massive quantities of minimally processed machine data for later retrieval and analysis. In general, storing minimally processed machine data and performing analysis operations at search time can provide greater flexibility because it enables an analyst to search all of the machine data, instead of searching only a pre-specified set of data items. This may enable an analyst to investigate different aspects of the machine data that previously were unavailable for analysis.

However, analyzing and searching massive quantities of machine data presents a number of challenges. For example, a data center, servers, or network appliances may generate many different types and formats of machine data (e.g., system logs, network packet data (e.g., wire data, etc.), sensor data, application program data, error logs, stack traces, system performance data, operating system data, virtualization data, etc.) from thousands of different components, which can collectively be very time-consuming to analyze. In another example, mobile devices may generate large amounts of information relating to data accesses, application performance, operating system performance, network performance, etc. There can be millions of mobile devices that report these types of information.

These challenges can be addressed by using an event-based data intake and query system, such as the SPLUNK® ENTERPRISE system developed by Splunk Inc. of San Francisco, Calif. The SPLUNK® ENTERPRISE system is the leading platform for providing real-time operational intelligence that enables organizations to collect, index, and search machine-generated data from various websites, applications, servers, networks, and mobile devices that power their businesses. The SPLUNK® ENTERPRISE system is particularly useful for analyzing data which is commonly found in system log files, network data, and other data input sources. Although many of the techniques described herein are explained with reference to a data intake and query system similar to the SPLUNK® ENTERPRISE system, these techniques are also applicable to other types of data systems.

In the SPLUNK® ENTERPRISE system, machine-generated data are collected and stored as "events". An event comprises a portion of the machine-generated data and is associated with a specific point in time. For example, events may be derived from "time series data," where the time series data comprises a sequence of data points (e.g., performance measurements from a computer system, etc.) that are associated with successive points in time. In general, each event can be associated with a timestamp that is derived from the raw data in the event, determined through interpolation between temporally proximate events having known timestamps, or determined based on other configurable rules for associating timestamps with events, etc.

In some instances, machine data can have a predefined format, where data items with specific data formats are stored at predefined locations in the data. For example, the machine data may include data stored as fields in a database table. In other instances, machine data may not have a predefined format, that is, the data is not at fixed, predefined locations, but the data does have repeatable patterns and is not random. This means that some machine data can comprise various data items of different data types and that may be stored at different locations within the data. For example, when the data source is an operating system log, an event can include one or more lines from the operating system log

containing raw data that includes different types of performance and diagnostic information associated with a specific point in time.

Examples of components which may generate machine data from which events can be derived include, but are not limited to, web servers, application servers, databases, firewalls, routers, operating systems, and software applications that execute on computer systems, mobile devices, sensors, Internet of Things (IoT) devices, etc. The data generated by such data sources can include, for example and without limitation, server log files, activity log files, configuration files, messages, network packet data, performance measurements, sensor measurements, etc.

The SPLUNK® ENTERPRISE system uses flexible schema to specify how to extract information from the event data. A flexible schema may be developed and redefined as needed. Note that a flexible schema may be applied to event data "on the fly," when it is needed (e.g., at search time, index time, ingestion time, etc.). When the schema is not applied to event data until search time it may be referred to as a "late-binding schema."

During operation, the SPLUNK® ENTERPRISE system starts with raw input data (e.g., one or more system logs, streams of network packet data, sensor data, application program data, error logs, stack traces, system performance data, etc.). The system divides this raw data into blocks (e.g., buckets of data, each associated with a specific time frame, etc.), and parses the raw data to produce timestamped events. The system stores the timestamped events in a data store. The system enables users to run queries against the stored data to, for example, retrieve events that meet criteria specified in a query, such as containing certain keywords or having specific values in defined fields. As used herein throughout, data that is part of an event is referred to as "event data". In this context, the term "field" refers to a location in the event data containing one or more values for a specific data item. As will be described in more detail herein, the fields are defined by extraction rules (e.g., regular expressions) that derive one or more values from the portion of raw machine data in each event that has a particular field specified by an extraction rule. The set of values so produced are semantically-related (such as IP address), even though the raw machine data in each event may be in different formats (e.g., semantically-related values may be in different positions in the events derived from different sources).

As noted above, the SPLUNK® ENTERPRISE system utilizes a late-binding schema to event data while performing queries on events. One aspect of a late-binding schema is applying "extraction rules" to event data to extract values for specific fields during search time. More specifically, the extraction rules for a field can include one or more instructions that specify how to extract a value for the field from the event data. An extraction rule can generally include any type of instruction for extracting values from data in events. In some cases, an extraction rule comprises a regular expression where a sequence of characters form a search pattern, in which case the rule is referred to as a "regex rule." The system applies the regex rule to the event data to extract values for associated fields in the event data by searching the event data for the sequence of characters defined in the regex rule.

In the SPLUNK® ENTERPRISE system, a field extractor may be configured to automatically generate extraction rules for certain field values in the events when the events are being created, indexed, or stored, or possibly at a later time. Alternatively, a user may manually define extraction rules for fields using a variety of techniques. In contrast to a conventional schema for a database system, a late-binding schema is not defined at data ingestion time. Instead, the late-binding schema can be developed on an ongoing basis until the time a query is actually executed. This means that extraction rules for the fields in a query may be provided in the query itself, or may be located during execution of the query. Hence, as a user learns more about the data in the events, the user can continue to refine the late-binding schema by adding new fields, deleting fields, or modifying the field extraction rules for use the next time the schema is used by the system. Because the SPLUNK® ENTERPRISE system maintains the underlying raw data and uses late-binding schema for searching the raw data, it enables a user to continue investigating and learn valuable insights about the raw data.

In some embodiments, a common field name may be used to reference two or more fields containing equivalent data items, even though the fields may be associated with different types of events that possibly have different data formats and different extraction rules. By enabling a common field name to be used to identify equivalent fields from different types of events generated by disparate data sources, the system facilitates use of a "common information model" (CIM) across the disparate data sources (further discussed with respect to FIG. **5**).

2.0. Operating Environment

FIG. **1** illustrates a networked computer system **100** in which an embodiment may be implemented. Those skilled in the art would understand that FIG. **1** represents one example of a networked computer system and other embodiments may use different arrangements.

The networked computer system **100** comprises one or more computing devices. These one or more computing devices comprise any combination of hardware and software configured to implement the various logical components described herein. For example, the one or more computing devices may include one or more memories that store instructions for implementing the various components described herein, one or more hardware processors configured to execute the instructions stored in the one or more memories, and various data repositories in the one or more memories for storing data structures utilized and manipulated by the various components.

In an embodiment, one or more client devices **102** are coupled to one or more host devices **106** and a data intake and query system **108** via one or more networks **104**. Networks **104** broadly represent one or more LANs, WANs, cellular networks (e.g., LTE, HSPA, 3G, and other cellular technologies), and/or networks using any of wired, wireless, terrestrial microwave, or satellite links, and may include the public Internet.

2.1. Host Devices

In the illustrated embodiment, a system **100** includes one or more host devices **106**. Host devices **106** may broadly include any number of computers, virtual machine instances, and/or data centers that are configured to host or execute one or more instances of host applications **114**. In general, a host device **106** may be involved, directly or indirectly, in processing requests received from client devices **102**. Each host device **106** may comprise, for example, one or more of a network device, a web server, an application server, a database server, etc. A collection of host devices **106** may be configured to implement a network-based service. For example, a provider of a network-based service may configure one or more host devices **106** and host applications

114 (e.g., one or more web servers, application servers, database servers, etc.) to collectively implement the network-based application.

In general, client devices **102** communicate with one or more host applications **114** to exchange information. The communication between a client device **102** and a host application **114** may, for example, be based on the Hypertext Transfer Protocol (HTTP) or any other network protocol. Content delivered from the host application **114** to a client device **102** may include, for example, HTML documents, media content, etc. The communication between a client device **102** and host application **114** may include sending various requests and receiving data packets. For example, in general, a client device **102** or application running on a client device may initiate communication with a host application **114** by making a request for a specific resource (e.g., based on an HTTP request), and the application server may respond with the requested content stored in one or more response packets.

In the illustrated embodiment, one or more of host applications **114** may generate various types of performance data during operation, including event logs, network data, sensor data, and other types of machine-generated data. For example, a host application **114** comprising a web server may generate one or more web server logs in which details of interactions between the web server and any number of client devices **102** is recorded. As another example, a host device **106** comprising a router may generate one or more router logs that record information related to network traffic managed by the router. As yet another example, a host application **114** comprising a database server may generate one or more logs that record information related to requests sent from other host applications **114** (e.g., web servers or application servers) for data managed by the database server.

2.2. Client Devices

Client devices **102** of FIG. **1** represent any computing device capable of interacting with one or more host devices **106** via a network **104**. Examples of client devices **102** may include, without limitation, smart phones, tablet computers, handheld computers, wearable devices, laptop computers, desktop computers, servers, portable media players, gaming devices, and so forth. In general, a client device **102** can provide access to different content, for instance, content provided by one or more host devices **106**, etc. Each client device **102** may comprise one or more client applications **110**, described in more detail in a separate section hereinafter.

2.3. Client Device Applications

In an embodiment, each client device **102** may host or execute one or more client applications **110** that are capable of interacting with one or more host devices **106** via one or more networks **104**. For instance, a client application **110** may be or comprise a web browser that a user may use to navigate to one or more websites or other resources provided by one or more host devices **106**. As another example, a client application **110** may comprise a mobile application or "app." For example, an operator of a network-based service hosted by one or more host devices **106** may make available one or more mobile apps that enable users of client devices **102** to access various resources of the network-based service. As yet another example, client applications **110** may include background processes that perform various operations without direct interaction from a user. A client application **110** may include a "plug-in" or "extension" to another application, such as a web browser plug-in or extension.

In an embodiment, a client application **110** may include a monitoring component **112**. At a high level, the monitoring component **112** comprises a software component or other logic that facilitates generating performance data related to a client device's operating state, including monitoring network traffic sent and received from the client device and collecting other device and/or application-specific information. Monitoring component **112** may be an integrated component of a client application **110**, a plug-in, an extension, or any other type of add-on component. Monitoring component **112** may also be a stand-alone process.

In one embodiment, a monitoring component **112** may be created when a client application **110** is developed, for example, by an application developer using a software development kit (SDK). The SDK may include custom monitoring code that can be incorporated into the code implementing a client application **110**. When the code is converted to an executable application, the custom code implementing the monitoring functionality can become part of the application itself.

In some cases, an SDK or other code for implementing the monitoring functionality may be offered by a provider of a data intake and query system, such as a system **108**. In such cases, the provider of the system **108** can implement the custom code so that performance data generated by the monitoring functionality is sent to the system **108** to facilitate analysis of the performance data by a developer of the client application or other users.

In an embodiment, the custom monitoring code may be incorporated into the code of a client application **110** in a number of different ways, such as the insertion of one or more lines in the client application code that call or otherwise invoke the monitoring component **112**. As such, a developer of a client application **110** can add one or more lines of code into the client application **110** to trigger the monitoring component **112** at desired points during execution of the application. Code that triggers the monitoring component may be referred to as a monitor trigger. For instance, a monitor trigger may be included at or near the beginning of the executable code of the client application **110** such that the monitoring component **112** is initiated or triggered as the application is launched, or included at other points in the code that correspond to various actions of the client application, such as sending a network request or displaying a particular interface.

In an embodiment, the monitoring component **112** may monitor one or more aspects of network traffic sent and/or received by a client application **110**. For example, the monitoring component **112** may be configured to monitor data packets transmitted to and/or from one or more host applications **114**. Incoming and/or outgoing data packets can be read or examined to identify network data contained within the packets, for example, and other aspects of data packets can be analyzed to determine a number of network performance statistics. Monitoring network traffic may enable information to be gathered particular to the network performance associated with a client application **110** or set of applications.

In an embodiment, network performance data refers to any type of data that indicates information about the network and/or network performance. Network performance data may include, for instance, a URL requested, a connection type (e.g., HTTP, HTTPS, etc.), a connection start time, a connection end time, an HTTP status code, request length, response length, request headers, response headers, connection status (e.g., completion, response time(s), failure, etc.), and the like. Upon obtaining network performance data

indicating performance of the network, the network performance data can be transmitted to a data intake and query system **108** for analysis.

Upon developing a client application **110** that incorporates a monitoring component **112**, the client application **110** can be distributed to client devices **102**. Applications generally can be distributed to client devices **102** in any manner, or they can be pre-loaded. In some cases, the application may be distributed to a client device **102** via an application marketplace or other application distribution system. For instance, an application marketplace or other application distribution system might distribute the application to a client device based on a request from the client device to download the application.

Examples of functionality that enables monitoring performance of a client device are described in U.S. patent application Ser. No. 14/524,748, entitled "UTILIZING PACKET HEADERS TO MONITOR NETWORK TRAFFIC IN ASSOCIATION WITH A CLIENT DEVICE", filed on 27 Oct. 2014, and which is hereby incorporated by reference in its entirety for all purposes.

In an embodiment, the monitoring component **112** may also monitor and collect performance data related to one or more aspects of the operational state of a client application **110** and/or client device **102**. For example, a monitoring component **112** may be configured to collect device performance information by monitoring one or more client device operations, or by making calls to an operating system and/or one or more other applications executing on a client device **102** for performance information. Device performance information may include, for instance, a current wireless signal strength of the device, a current connection type and network carrier, current memory performance information, a geographic location of the device, a device orientation, and any other information related to the operational state of the client device.

In an embodiment, the monitoring component **112** may also monitor and collect other device profile information including, for example, a type of client device, a manufacturer and model of the device, versions of various software applications installed on the device, and so forth.

In general, a monitoring component **112** may be configured to generate performance data in response to a monitor trigger in the code of a client application **110** or other triggering application event, as described above, and to store the performance data in one or more data records. Each data record, for example, may include a collection of field-value pairs, each field-value pair storing a particular item of performance data in association with a field for the item. For example, a data record generated by a monitoring component **112** may include a "networkLatency" field (not shown in the Figure) in which a value is stored. This field indicates a network latency measurement associated with one or more network requests. The data record may include a "state" field to store a value indicating a state of a network connection, and so forth for any number of aspects of collected performance data.

2.4. Data Server System

FIG. **2** depicts a block diagram of an exemplary data intake and query system **108**, similar to the SPLUNK® ENTERPRISE system. System **108** includes one or more forwarders **204** that receive data from a variety of input data sources **202**, and one or more indexers **206** that process and store the data in one or more data stores **208**. These forwarders and indexers can comprise separate computer systems, or may alternatively comprise separate processes executing on one or more computer systems.

Each data source **202** broadly represents a distinct source of data that can be consumed by a system **108**. Examples of a data source **202** include, without limitation, data files, directories of files, data sent over a network, event logs, registries, etc.

During operation, the forwarders **204** identify which indexers **206** receive data collected from a data source **202** and forward the data to the appropriate indexers. Forwarders **204** can also perform operations on the data before forwarding, including removing extraneous data, detecting timestamps in the data, parsing data, indexing data, routing data based on criteria relating to the data being routed, and/or performing other data transformations.

In an embodiment, a forwarder **204** may comprise a service accessible to client devices **102** and host devices **106** via a network **104**. For example, one type of forwarder **204** may be capable of consuming vast amounts of real-time data from a potentially large number of client devices **102** and/or host devices **106**. The forwarder **204** may, for example, comprise a computing device which implements multiple data pipelines or "queues" to handle forwarding of network data to indexers **206**. A forwarder **204** may also perform many of the functions that are performed by an indexer. For example, a forwarder **204** may perform keyword extractions on raw data or parse raw data to create events. A forwarder **204** may generate time stamps for events. Additionally or alternatively, a forwarder **204** may perform routing of events to indexers. Data store **208** may contain events derived from machine data from a variety of sources all pertaining to the same component in an IT environment, and this data may be produced by the machine in question or by other components in the IT environment.

2.5. Data Ingestion

FIG. **3** depicts a flow chart illustrating an example data flow performed by Data Intake and Query system **108**, in accordance with the disclosed embodiments. The data flow illustrated in FIG. **3** is provided for illustrative purposes only; those skilled in the art would understand that one or more of the steps of the processes illustrated in FIG. **3** may be removed or the ordering of the steps may be changed. Furthermore, for the purposes of illustrating a clear example, one or more particular system components are described in the context of performing various operations during each of the data flow stages. For example, a forwarder is described as receiving and processing data during an input phase; an indexer is described as parsing and indexing data during parsing and indexing phases; and a search head is described as performing a search query during a search phase. However, other system arrangements and distributions of the processing steps across system components may be used.

2.5.1. Input

At block **302**, a forwarder receives data from an input source, such as a data source **202** shown in FIG. **2**. A forwarder initially may receive the data as a raw data stream generated by the input source. For example, a forwarder may receive a data stream from a log file generated by an application server, from a stream of network data from a network device, or from any other source of data. In one embodiment, a forwarder receives the raw data and may segment the data stream into "blocks", or "buckets," possibly of a uniform data size, to facilitate subsequent processing steps.

At block **304**, a forwarder or other system component annotates each block generated from the raw data with one or more metadata fields. These metadata fields may, for example, provide information related to the data block as a whole and may apply to each event that is subsequently

derived from the data in the data block. For example, the metadata fields may include separate fields specifying each of a host, a source, and a source type related to the data block. A host field may contain a value identifying a host name or IP address of a device that generated the data. A source field may contain a value identifying a source of the data, such as a pathname of a file or a protocol and port related to received network data. A source type field may contain a value specifying a particular source type label for the data. Additional metadata fields may also be included during the input phase, such as a character encoding of the data, if known, and possibly other values that provide information relevant to later processing steps. In an embodiment, a forwarder forwards the annotated data blocks to another system component (typically an indexer) for further processing.

The SPLUNK® ENTERPRISE system allows forwarding of data from one SPLUNK® ENTERPRISE instance to another, or even to a third-party system. SPLUNK® ENTERPRISE system can employ different types of forwarders in a configuration.

In an embodiment, a forwarder may contain the essential components needed to forward data. It can gather data from a variety of inputs and forward the data to a SPLUNK® ENTERPRISE server for indexing and searching. It also can tag metadata (e.g., source, source type, host, etc.).

Additionally or optionally, in an embodiment, a forwarder has the capabilities of the aforementioned forwarder as well as additional capabilities. The forwarder can parse data before forwarding the data (e.g., associate a time stamp with a portion of data and create an event, etc.) and can route data based on criteria such as source or type of event. It can also index data locally while forwarding the data to another indexer.

2.5.2. Parsing

At block **306**, an indexer receives data blocks from a forwarder and parses the data to organize the data into events. In an embodiment, to organize the data into events, an indexer may determine a source type associated with each data block (e.g., by extracting a source type label from the metadata fields associated with the data block, etc.) and refer to a source type configuration corresponding to the identified source type. The source type definition may include one or more properties that indicate to the indexer to automatically determine the boundaries of events within the data. In general, these properties may include regular expression-based rules or delimiter rules where, for example, event boundaries may be indicated by predefined characters or character strings. These predefined characters may include punctuation marks or other special characters including, for example, carriage returns, tabs, spaces, line breaks, etc. If a source type for the data is unknown to the indexer, an indexer may infer a source type for the data by examining the structure of the data. Then, it can apply an inferred source type definition to the data to create the events.

At block **308**, the indexer determines a timestamp for each event. Similar to the process for creating events, an indexer may again refer to a source type definition associated with the data to locate one or more properties that indicate instructions for determining a timestamp for each event. The properties may, for example, instruct an indexer to extract a time value from a portion of data in the event, to interpolate time values based on timestamps associated with temporally proximate events, to create a timestamp based on a time the event data was received or generated, to use the timestamp of a previous event, or use any other rules for determining timestamps.

At block **310**, the indexer associates with each event one or more metadata fields including a field containing the timestamp (in some embodiments, a timestamp may be included in the metadata fields) determined for the event. These metadata fields may include a number of "default fields" that are associated with all events, and may also include one more custom fields as defined by a user. Similar to the metadata fields associated with the data blocks at block **304**, the default metadata fields associated with each event may include a host, source, and source type field including or in addition to a field storing the timestamp.

At block **312**, an indexer may optionally apply one or more transformations to data included in the events created at block **306**. For example, such transformations can include removing a portion of an event (e.g., a portion used to define event boundaries, extraneous characters from the event, other extraneous text, etc.), masking a portion of an event (e.g., masking a credit card number), removing redundant portions of an event, etc. The transformations applied to event data may, for example, be specified in one or more configuration files and referenced by one or more source type definitions.

2.5.3. Indexing

At blocks **314** and **316**, an indexer can optionally generate a keyword index to facilitate fast keyword searching for event data. To build a keyword index, at block **314**, the indexer identifies a set of keywords in each event. At block **316**, the indexer includes the identified keywords in an index, which associates each stored keyword with reference pointers to events containing that keyword (or to locations within events where that keyword is located, other location identifiers, etc.). When an indexer subsequently receives a keyword-based query, the indexer can access the keyword index to quickly identify events containing the keyword.

In some embodiments, the keyword index may include entries for name-value pairs found in events, where a name-value pair can include a pair of keywords connected by a symbol, such as an equals sign or colon. This way, events containing these name-value pairs can be quickly located. In some embodiments, fields can automatically be generated for some or all of the name-value pairs at the time of indexing. For example, if the string "dest=10.0.1.2" is found in an event, a field named "dest" may be created for the event, and assigned a value of "10.0.1.2".

At block **318**, the indexer stores the events with an associated timestamp in a data store **208**. Timestamps enable a user to search for events based on a time range. In one embodiment, the stored events are organized into "buckets," where each bucket stores events associated with a specific time range based on the timestamps associated with each event. This may not only improve time-based searching, but also allows for events with recent timestamps, which may have a higher likelihood of being accessed, to be stored in a faster memory to facilitate faster retrieval. For example, buckets containing the most recent events can be stored in flash memory rather than on a hard disk.

Each indexer **206** may be responsible for storing and searching a subset of the events contained in a corresponding data store **208**. By distributing events among the indexers and data stores, the indexers can analyze events for a query in parallel. For example, using map-reduce techniques, each indexer returns partial responses for a subset of events to a search head that combines the results to produce an answer for the query. By storing events in buckets for specific time ranges, an indexer may further optimize data retrieval process by searching buckets corresponding to time ranges that are relevant to a query.

Moreover, events and buckets can also be replicated across different indexers and data stores to facilitate high availability and disaster recovery as described in U.S. patent application Ser. No. 14/266,812, entitled "SITE-BASED SEARCH AFFINITY", filed on 30 Apr. 2014, and in U.S. patent application Ser. No. 14/266,817, entitled "MULTI-SITE CLUSTERING", also filed on 30 Apr. 2014, each of which is hereby incorporated by reference in its entirety for all purposes.

2.6. Query Processing

FIG. **4** is a flow diagram that illustrates an exemplary process that a search head and one or more indexers may perform during a search query. At block **402**, a search head receives a search query from a client. At block **404**, the search head analyzes the search query to determine what portion(s) of the query can be delegated to indexers and what portions of the query can be executed locally by the search head. At block **406**, the search head distributes the determined portions of the query to the appropriate indexers. In an embodiment, a search head cluster may take the place of an independent search head where each search head in the search head cluster coordinates with peer search heads in the search head cluster to schedule jobs, replicate search results, update configurations, fulfill search requests, etc. In an embodiment, the search head (or each search head) communicates with a master node (also known as a cluster master, not shown in FIG.) that provides the search head with a list of indexers to which the search head can distribute the determined portions of the query. The master node maintains a list of active indexers and can also designate which indexers may have responsibility for responding to queries over certain sets of events. A search head may communicate with the master node before the search head distributes queries to indexers to discover the addresses of active indexers.

At block **408**, the indexers to which the query was distributed, search data stores associated with them for events that are responsive to the query. To determine which events are responsive to the query, the indexer searches for events that match the criteria specified in the query. These criteria can include matching keywords or specific values for certain fields. The searching operations at block **408** may use the late-binding schema to extract values for specified fields from events at the time the query is processed. In an embodiment, one or more rules for extracting field values may be specified as part of a source type definition. The indexers may then either send the relevant events back to the search head, or use the events to determine a partial result, and send the partial result back to the search head.

At block **410**, the search head combines the partial results and/or events received from the indexers to produce a final result for the query. This final result may comprise different types of data depending on what the query requested. For example, the results can include a listing of matching events returned by the query, or some type of visualization of the data from the returned events. In another example, the final result can include one or more calculated values derived from the matching events.

The results generated by the system **108** can be returned to a client using different techniques. For example, one technique streams results or relevant events back to a client in real-time as they are identified. Another technique waits to report the results to the client until a complete set of results (which may include a set of relevant events or a result based on relevant events) is ready to return to the client. Yet another technique streams interim results or relevant events back to the client in real-time until a complete set of results is ready, and then returns the complete set of results to the client. In another technique, certain results are stored as "search jobs" and the client may retrieve the results by referring the search jobs.

The search head can also perform various operations to make the search more efficient. For example, before the search head begins execution of a query, the search head can determine a time range for the query and a set of common keywords that all matching events include. The search head may then use these parameters to query the indexers to obtain a superset of the eventual results. Then, during a filtering stage, the search head can perform field-extraction operations on the superset to produce a reduced set of search results. This speeds up queries that are performed on a periodic basis.

2.7. Field Extraction

The search head **210** allows users to search and visualize event data extracted from raw machine data received from homogenous data sources. It also allows users to search and visualize event data extracted from raw machine data received from heterogeneous data sources. The search head **210** includes various mechanisms, which may additionally reside in an indexer **206**, for processing a query. Splunk Processing Language (SPL), used in conjunction with the SPLUNK® ENTERPRISE system, can be utilized to make a query. SPL is a pipelined search language in which a set of inputs is operated on by a first command in a command line, and then a subsequent command following the pipe symbol "|" operates on the results produced by the first command, and so on for additional commands. Other query languages, such as the Structured Query Language ("SQL"), can be used to create a query.

In response to receiving the search query, search head **210** uses extraction rules to extract values for the fields associated with a field or fields in the event data being searched. The search head **210** obtains extraction rules that specify how to extract a value for certain fields from an event. Extraction rules can comprise regex rules that specify how to extract values for the relevant fields. In addition to specifying how to extract field values, the extraction rules may also include instructions for deriving a field value by performing a function on a character string or value retrieved by the extraction rule. For example, a transformation rule may truncate a character string, or convert the character string into a different data format. In some cases, the query itself can specify one or more extraction rules.

The search head **210** can apply the extraction rules to event data that it receives from indexers **206**. Indexers **206** may apply the extraction rules to events in an associated data store **208**. Extraction rules can be applied to all the events in a data store, or to a subset of the events that have been filtered based on some criteria (e.g., event time stamp values, etc.). Extraction rules can be used to extract one or more values for a field from events by parsing the event data and examining the event data for one or more patterns of characters, numbers, delimiters, etc., that indicate where the field begins and, optionally, ends.

FIG. **5** illustrates an example of raw machine data received from disparate data sources. In this example, a user submits an order for merchandise using a vendor's shopping application program **501** running on the user's system. In this example, the order was not delivered to the vendor's server due to a resource exception at the destination server that is detected by the middleware code **502**. The user then sends a message to the customer support **503** to complain about the order failing to complete. The three systems **501**, **502**, and **503** are disparate systems that do not have a

common logging format. The order application **501** sends log data **504** to the SPLUNK® ENTERPRISE system in one format, the middleware code **502** sends error log data **505** in a second format, and the support server **503** sends log data **506** in a third format.

Using the log data received at one or more indexers **206** from the three systems the vendor can uniquely obtain an insight into user activity, user experience, and system behavior. The search head **210** allows the vendor's administrator to search the log data from the three systems that one or more indexers **206** are responsible for searching, thereby obtaining correlated information, such as the order number and corresponding customer ID number of the person placing the order. The system also allows the administrator to see a visualization of related events via a user interface. The administrator can query the search head **210** for customer ID field value matches across the log data from the three systems that are stored at the one or more indexers **206**. The customer ID field value exists in the data gathered from the three systems, but the customer ID field value may be located in different areas of the data given differences in the architecture of the systems—there is a semantic relationship between the customer ID field values generated by the three systems. The search head **210** requests event data from the one or more indexers **206** to gather relevant event data from the three systems. It then applies extraction rules to the event data in order to extract field values that it can correlate. The search head may apply a different extraction rule to each set of events from each system when the event data format differs among systems. In this example, the user interface can display to the administrator the event data corresponding to the common customer ID field values **507**, **508**, and **509**, thereby providing the administrator with insight into a customer's experience.

Note that query results can be returned to a client, a search head, or any other system component for further processing. In general, query results may include a set of one or more events, a set of one or more values obtained from the events, a subset of the values, statistics calculated based on the values, a report containing the values, or a visualization, such as a graph or chart, generated from the values.

2.8. Example Search Screen

FIG. **6**A illustrates an example search screen **600** in accordance with the disclosed embodiments. Search screen **600** includes a search bar **602** that accepts user input in the form of a search string. It also includes a time range picker **612** that enables the user to specify a time range for the search. For "historical searches" the user can select a specific time range, or alternatively a relative time range, such as "today," "yesterday" or "last week." For "real-time searches," the user can select the size of a preceding time window to search for real-time events. Search screen **600** also initially displays a "data summary" dialog as is illustrated in FIG. **6**B that enables the user to select different sources for the event data, such as by selecting specific hosts and log files.

After the search is executed, the search screen **600** in FIG. **6**A can display the results through search results tabs **604**, wherein search results tabs **604** includes: an "events tab" that displays various information about events returned by the search; a "statistics tab" that displays statistics about the search results; and a "visualization tab" that displays various visualizations of the search results. The events tab illustrated in FIG. **6**A displays a timeline graph **605** that graphically illustrates the number of events that occurred in one-hour intervals over the selected time range. It also displays an events list **608** that enables a user to view the raw data in each of the returned events. It additionally displays a fields sidebar **606** that includes statistics about occurrences of specific fields in the returned events, including "selected fields" that are pre-selected by the user, and "interesting fields" that are automatically selected by the system based on pre-specified criteria.

2.9. Data Models

A data model is a hierarchically structured search-time mapping of semantic knowledge about one or more datasets. It encodes the domain knowledge necessary to build a variety of specialized searches of those datasets. Those searches, in turn, can be used to generate reports.

A data model is composed of one or more "objects" (or "data model objects") that define or otherwise correspond to a specific set of data.

Objects in data models can be arranged hierarchically in parent/child relationships. Each child object represents a subset of the dataset covered by its parent object. The top-level objects in data models are collectively referred to as "root objects."

Child objects have inheritance. Data model objects are defined by characteristics that mostly break down into constraints and attributes. Child objects inherit constraints and attributes from their parent objects and have additional constraints and attributes of their own. Child objects provide a way of filtering events from parent objects. Because a child object always provides an additional constraint in addition to the constraints it has inherited from its parent object, the dataset it represents is always a subset of the dataset that its parent represents.

For example, a first data model object may define a broad set of data pertaining to e-mail activity generally, and another data model object may define specific datasets within the broad dataset, such as a subset of the e-mail data pertaining specifically to e-mails sent. Examples of data models can include electronic mail, authentication, databases, intrusion detection, malware, application state, alerts, compute inventory, network sessions, network traffic, performance, audits, updates, vulnerabilities, etc. Data models and their objects can be designed by knowledge managers in an organization, and they can enable downstream users to quickly focus on a specific set of data. For example, a user can simply select an "e-mail activity" data model object to access a dataset relating to e-mails generally (e.g., sent or received), or select an "e-mails sent" data model object (or data sub-model object) to access a dataset relating to e-mails sent.

A data model object may be defined by (1) a set of search constraints, and (2) a set of fields. Thus, a data model object can be used to quickly search data to identify a set of events and to identify a set of fields to be associated with the set of events. For example, an "e-mails sent" data model object may specify a search for events relating to e-mails that have been sent, and specify a set of fields that are associated with the events. Thus, a user can retrieve and use the "e-mails sent" data model object to quickly search source data for events relating to sent e-mails, and may be provided with a listing of the set of fields relevant to the events in a user interface screen.

A child of the parent data model may be defined by a search (typically a narrower search) that produces a subset of the events that would be produced by the parent data model's search. The child's set of fields can include a subset of the set of fields of the parent data model and/or additional fields. Data model objects that reference the subsets can be arranged in a hierarchical manner, so that child subsets of events are proper subsets of their parents. A user iteratively

applies a model development tool (not shown in FIG.) to prepare a query that defines a subset of events and assigns an object name to that subset. A child subset is created by further limiting a query that generated a parent subset. A late-binding schema of field extraction rules is associated with each object or subset in the data model.

Data definitions in associated schemas can be taken from the common information model (CIM) or can be devised for a particular schema and optionally added to the CIM. Child objects inherit fields from parents and can include fields not present in parents. A model developer can select fewer extraction rules than are available for the sources returned by the query that defines events belonging to a model. Selecting a limited set of extraction rules can be a tool for simplifying and focusing the data model, while allowing a user flexibility to explore the data subset. Development of a data model is further explained in U.S. Pat. Nos. 8,788,525 and 8,788,526, both entitled "DATA MODEL FOR MACHINE DATA FOR SEMANTIC SEARCH", both issued on 22 Jul. 2014, U.S. Pat. No. 8,983,994, entitled "GENERATION OF A DATA MODEL FOR SEARCHING MACHINE DATA", issued on 17 Mar. 2015, U.S. patent application Ser. No. 14/611,232, entitled "GENERATION OF A DATA MODEL APPLIED TO QUERIES", filed on 31 Jan. 2015, and U.S. patent application Ser. No. 14/815,884, entitled "GENERATION OF A DATA MODEL APPLIED TO OBJECT QUERIES", filed on 31 Jul. 2015, each of which is hereby incorporated by reference in its entirety for all purposes. See, also, Knowledge Manager Manual, Build a Data Model, Splunk Enterprise 6.1.3 pp. 150-204 (Aug. 25, 2014).

A data model can also include reports. One or more report formats can be associated with a particular data model and be made available to run against the data model. A user can use child objects to design reports with object datasets that already have extraneous data pre-filtered out. In an embodiment, the data intake and query system **108** provides the user with the ability to produce reports (e.g., a table, chart, visualization, etc.) without having to enter SPL, SQL, or other query language terms into a search screen. Data models are used as the basis for the search feature.

Data models may be selected in a report generation interface. The report generator supports drag-and-drop organization of fields to be summarized in a report. When a model is selected, the fields with available extraction rules are made available for use in the report. The user may refine and/or filter search results to produce more precise reports. The user may select some fields for organizing the report and select other fields for providing detail according to the report organization. For example, "region" and "salesperson" are fields used for organizing the report and sales data can be summarized (subtotaled and totaled) within this organization. The report generator allows the user to specify one or more fields within events and apply statistical analysis on values extracted from the specified one or more fields. The report generator may aggregate search results across sets of events and generate statistics based on aggregated search results. Building reports using the report generation interface is further explained in U.S. patent application Ser. No. 14/503,335, entitled "GENERATING REPORTS FROM UNSTRUCTURED DATA", filed on 30 Sep. 2014, and which is hereby incorporated by reference in its entirety for all purposes, and in Pivot Manual, Splunk Enterprise 6.1.3 (Aug. 4, 2014). Data visualizations also can be generated in a variety of formats, by reference to the data model. Reports, data visualizations, and data model objects can be saved and associated with the data model for future use. The data model object may be used to perform searches of other data.

FIGS. **12**, **13**, and **7**A-7D illustrate a series of user interface screens where a user may select report generation options using data models. The report generation process may be driven by a predefined data model object, such as a data model object defined and/or saved via a reporting application or a data model object obtained from another source. A user can load a saved data model object using a report editor. For example, the initial search query and fields used to drive the report editor may be obtained from a data model object. The data model object that is used to drive a report generation process may define a search and a set of fields. Upon loading of the data model object, the report generation process may enable a user to use the fields (e.g., the fields defined by the data model object) to define criteria for a report (e.g., filters, split rows/columns, aggregates, etc.) and the search may be used to identify events (e.g., to identify events responsive to the search) used to generate the report. That is, for example, if a data model object is selected to drive a report editor, the graphical user interface of the report editor may enable a user to define reporting criteria for the report using the fields associated with the selected data model object, and the events used to generate the report may be constrained to the events that match, or otherwise satisfy, the search constraints of the selected data model object.

The selection of a data model object for use in driving a report generation may be facilitated by a data model object selection interface. FIG. **12** illustrates an example interactive data model selection graphical user interface **1200** of a report editor that displays a listing of available data models **1201**. The user may select one of the data models **1202**.

FIG. **13** illustrates an example data model object selection graphical user interface **1300** that displays available data objects **1301** for the selected data object model **1202**. The user may select one of the displayed data model objects **1302** for use in driving the report generation process.

Once a data model object is selected by the user, a user interface screen **700** shown in FIG. **7**A may display an interactive listing of automatic field identification options **701** based on the selected data model object. For example, a user may select one of the three illustrated options (e.g., the "All Fields" option **702**, the "Selected Fields" option **703**, or the "Coverage" option (e.g., fields with at least a specified % of coverage) **704**). If the user selects the "All Fields" option **702**, all of the fields identified from the events that were returned in response to an initial search query may be selected. That is, for example, all of the fields of the identified data model object fields may be selected. If the user selects the "Selected Fields" option **703**, only the fields from the fields of the identified data model object fields that are selected by the user may be used. If the user selects the "Coverage" option **704**, only the fields of the identified data model object fields meeting a specified coverage criteria may be selected. A percent coverage may refer to the percentage of events returned by the initial search query that a given field appears in. Thus, for example, if an object dataset includes 10,000 events returned in response to an initial search query, and the "avg_age" field appears in 854 of those 10,000 events, then the "avg_age" field would have a coverage of 8.54% for that object dataset. If, for example, the user selects the "Coverage" option and specifies a coverage value of 2%, only fields having a coverage value equal to or greater than 2% may be selected. The number of fields corresponding to each selectable option may be displayed in association with each option. For example, "97"

displayed next to the "All Fields" option **702** indicates that 97 fields will be selected if the "All Fields" option is selected. The "3" displayed next to the "Selected Fields" option **703** indicates that 3 of the 97 fields will be selected if the "Selected Fields" option is selected. The "49" displayed next to the "Coverage" option **704** indicates that 49 of the 97 fields (e.g., the 49 fields having a coverage of 2% or greater) will be selected if the "Coverage" option is selected. The number of fields corresponding to the "Coverage" option may be dynamically updated based on the specified percent of coverage.

FIG. **7**B illustrates an example graphical user interface screen (also called the pivot interface) **705** displaying the reporting application's "Report Editor" page. The screen may display interactive elements for defining various elements of a report. For example, the page includes a "Filters" element **706**, a "Split Rows" element **707**, a "Split Columns" element **708**, and a "Column Values" element **709**. The page may include a list of search results **711**. In this example, the Split Rows element **707** is expanded, revealing a listing of fields **710** that can be used to define additional criteria (e.g., reporting criteria). The listing of fields **710** may correspond to the selected fields (attributes). That is, the listing of fields **710** may list only the fields previously selected, either automatically and/or manually by a user. FIG. **7**C illustrates a formatting dialogue **712** that may be displayed upon selecting a field from the listing of fields **710**. The dialogue can be used to format the display of the results of the selection (e.g., label the column to be displayed as "component").

FIG. **7**D illustrates an example graphical user interface screen **705** including a table of results **713** based on the selected criteria including splitting the rows by the "component" field. A column **714** having an associated count for each component listed in the table may be displayed that indicates an aggregate count of the number of times that the particular field-value pair (e.g., the value in a row) occurs in the set of events responsive to the initial search query.

FIG. **14** illustrates an example graphical user interface screen **1400** that allows the user to filter search results and to perform statistical analysis on values extracted from specific fields in the set of events. In this example, the top ten product names ranked by price are selected as a filter **1401** that causes the display of the ten most popular products sorted by price. Each row is displayed by product name and price **1402**. This results in each product displayed in a column labeled "product name" along with an associated price in a column labeled "price" **1406**. Statistical analysis of other fields in the events associated with the ten most popular products have been specified as column values **1403**. A count of the number of successful purchases for each product is displayed in column **1404**. This statistics may be produced by filtering the search results by the product name, finding all occurrences of a successful purchase in a field within the events and generating a total of the number of occurrences. A sum of the total sales is displayed in column **1405**, which is a result of the multiplication of the price and the number of successful purchases for each product.

The reporting application allows the user to create graphical visualizations of the statistics generated for a report. For example, FIG. **15** illustrates an example graphical user interface **1500** that displays a set of components and associated statistics **1501**. The reporting application allows the user to select a visualization of the statistics in a graph (e.g., bar chart, scatter plot, area chart, line chart, pie chart, radial gauge, marker gauge, filler gauge, etc.). FIG. **16** illustrates an example of a bar chart visualization **1600** of an aspect of the statistical data **1501**. FIG. **17** illustrates a scatter plot visualization **1700** of an aspect of the statistical data **1501**.

2.10. Acceleration Technique

The above-described system provides significant flexibility by enabling a user to analyze massive quantities of minimally processed data "on the fly" at search time instead of storing pre-specified portions of the data in a database at ingestion time. This flexibility enables a user to see valuable insights, correlate data, and perform subsequent queries to examine interesting aspects of the data that may not have been apparent at ingestion time.

However, performing extraction and analysis operations at search time can involve a large amount of data and require a large number of computational operations, which can cause delays in processing the queries. Advantageously, SPLUNK® ENTERPRISE system employs a number of unique acceleration techniques that have been developed to speed up analysis operations performed at search time. These techniques include: (1) performing search operations in parallel across multiple indexers; (2) using a keyword index; (3) using a high performance analytics store; and (4) accelerating the process of generating reports. These novel techniques are described in more detail below.

2.10.1. Aggregation Technique

To facilitate faster query processing, a query can be structured such that multiple indexers perform the query in parallel, while aggregation of search results from the multiple indexers is performed locally at the search head. For example, FIG. **8** illustrates how a search query **802** received from a client at a search head **210** can split into two phases, including: (1) subtasks **804** (e.g., data retrieval or simple filtering) that may be performed in parallel by indexers **206** for execution, and (2) a search results aggregation operation **806** to be executed by the search head when the results are ultimately collected from the indexers.

During operation, upon receiving search query **802**, a search head **210** determines that a portion of the operations involved with the search query may be performed locally by the search head. The search head modifies search query **802** by substituting "stats" (create aggregate statistics over results sets received from the indexers at the search head) with "prestats" (create statistics by the indexer from local results set) to produce search query **804**, and then distributes search query **804** to distributed indexers, which are also referred to as "search peers." Note that search queries may generally specify search criteria or operations to be performed on events that meet the search criteria. Search queries may also specify field names, as well as search criteria for the values in the fields or operations to be performed on the values in the fields. Moreover, the search head may distribute the full search query to the search peers as illustrated in FIG. **4**, or may alternatively distribute a modified version (e.g., a more restricted version) of the search query to the search peers. In this example, the indexers are responsible for producing the results and sending them to the search head. After the indexers return the results to the search head, the search head aggregates the received results **806** to form a single search result set. By executing the query in this manner, the system effectively distributes the computational operations across the indexers while minimizing data transfers.

2.10.2. Keyword Index

As described above with reference to the flow charts in FIG. **3** and FIG. **4**, data intake and query system **108** can construct and maintain one or more keyword indices to quickly identify events containing specific keywords. This

technique can greatly speed up the processing of queries involving specific keywords. As mentioned above, to build a keyword index, an indexer first identifies a set of keywords. Then, the indexer includes the identified keywords in an index, which associates each stored keyword with references to events containing that keyword, or to locations within events where that keyword is located. When an indexer subsequently receives a keyword-based query, the indexer can access the keyword index to quickly identify events containing the keyword.

2.10.3. High Performance Analytics Store

To speed up certain types of queries, some embodiments of system **108** create a high performance analytics store, which is referred to as a "summarization table," that contains entries for specific field-value pairs. Each of these entries keeps track of instances of a specific value in a specific field in the event data and includes references to events containing the specific value in the specific field. For example, an example entry in a summarization table can keep track of occurrences of the value "94107" in a "ZIP code" field of a set of events and the entry includes references to all of the events that contain the value "94107" in the ZIP code field. This optimization technique enables the system to quickly process queries that seek to determine how many events have a particular value for a particular field. To this end, the system can examine the entry in the summarization table to count instances of the specific value in the field without having to go through the individual events or perform data extractions at search time. Also, if the system needs to process all events that have a specific field-value combination, the system can use the references in the summarization table entry to directly access the events to extract further information without having to search all of the events to find the specific field-value combination at search time.

In some embodiments, the system maintains a separate summarization table for each of the above-described time-specific buckets that stores events for a specific time range. A bucket-specific summarization table includes entries for specific field-value combinations that occur in events in the specific bucket. Alternatively, the system can maintain a separate summarization table for each indexer. The indexer-specific summarization table includes entries for the events in a data store that are managed by the specific indexer. Indexer-specific summarization tables may also be bucket-specific.

The summarization table can be populated by running a periodic query that scans a set of events to find instances of a specific field-value combination, or alternatively instances of all field-value combinations for a specific field. A periodic query can be initiated by a user, or can be scheduled to occur automatically at specific time intervals. A periodic query can also be automatically launched in response to a query that asks for a specific field-value combination.

In some cases, when the summarization tables may not cover all of the events that are relevant to a query, the system can use the summarization tables to obtain partial results for the events that are covered by summarization tables, but may also have to search through other events that are not covered by the summarization tables to produce additional results. These additional results can then be combined with the partial results to produce a final set of results for the query. The summarization table and associated techniques are described in more detail in U.S. Pat. No. 8,682,925, entitled "DISTRIBUTED HIGH PERFORMANCE ANALYTICS STORE", issued on 25 Mar. 2014, U.S. patent application Ser. No. 14/170,159, entitled "SUPPLEMENTING A HIGH PERFORMANCE ANALYTICS STORE WITH EVALUATION OF INDIVIDUAL EVENTS TO RESPOND TO AN EVENT QUERY", filed on 31 Jan. 2014, and U.S. patent application Ser. No. 14/815,973, entitled "STORAGE MEDIUM AND CONTROL DEVICE", filed on 21 Feb. 2014, each of which is hereby incorporated by reference in its entirety.

2.10.4. Accelerating Report Generation

In some embodiments, a data server system such as the SPLUNK® ENTERPRISE system can accelerate the process of periodically generating updated reports based on query results. To accelerate this process, a summarization engine automatically examines the query to determine whether generation of updated reports can be accelerated by creating intermediate summaries. If reports can be accelerated, the summarization engine periodically generates a summary covering data obtained during a latest non-overlapping time period. For example, where the query seeks events meeting a specified criteria, a summary for the time period includes only events within the time period that meet the specified criteria. Similarly, if the query seeks statistics calculated from the events, such as the number of events that match the specified criteria, then the summary for the time period includes the number of events in the period that match the specified criteria.

In addition to the creation of the summaries, the summarization engine schedules the periodic updating of the report associated with the query. During each scheduled report update, the query engine determines whether intermediate summaries have been generated covering portions of the time period covered by the report update. If so, then the report is generated based on the information contained in the summaries. Also, if additional event data has been received and has not yet been summarized, and is required to generate the complete report, the query can be run on this additional event data. Then, the results returned by this query on the additional event data, along with the partial results obtained from the intermediate summaries, can be combined to generate the updated report. This process is repeated each time the report is updated. Alternatively, if the system stores events in buckets covering specific time ranges, then the summaries can be generated on a bucket-by-bucket basis. Note that producing intermediate summaries can save the work involved in re-running the query for previous time periods, so advantageously only the newer event data needs to be processed while generating an updated report. These report acceleration techniques are described in more detail in U.S. Pat. No. 8,589,403, entitled "COMPRESSED JOURNALING IN EVENT TRACKING FILES FOR METADATA RECOVERY AND REPLICATION", issued on 19 Nov. 2013, U.S. Pat. No. 8,412,696, entitled "REAL TIME SEARCHING AND REPORTING", issued on 2 Apr. 2011, and U.S. Pat. Nos. 8,589,375 and 8,589,432, both also entitled "REAL TIME SEARCHING AND REPORTING", both issued on 19 Nov. 2013, each of which is hereby incorporated by reference in its entirety.

2.11. Security Features

The SPLUNK® ENTERPRISE platform provides various schemas, dashboards and visualizations that simplify developers' task to create applications with additional capabilities. One such application is the SPLUNK® APP FOR ENTERPRISE SECURITY, which performs monitoring and alerting operations and includes analytics to facilitate identifying both known and unknown security threats based on large volumes of data stored by the SPLUNK® ENTERPRISE system. SPLUNK® APP FOR ENTERPRISE SECURITY provides the security practitioner with visibility into security-relevant threats found in the enterprise infra-

structure by capturing, monitoring, and reporting on data from enterprise security devices, systems, and applications. Through the use of SPLUNK® ENTERPRISE searching and reporting capabilities, SPLUNK® APP FOR ENTERPRISE SECURITY provides a top-down and bottom-up view of an organization's security posture.

The SPLUNK® APP FOR ENTERPRISE SECURITY leverages SPLUNK® ENTERPRISE search-time normalization techniques, saved searches, and correlation searches to provide visibility into security-relevant threats and activity and generate notable events for tracking. The App enables the security practitioner to investigate and explore the data to find new or unknown threats that do not follow signature-based patterns.

Conventional Security Information and Event Management (SIEM) systems that lack the infrastructure to effectively store and analyze large volumes of security-related data. Traditional SIEM systems typically use fixed schemas to extract data from pre-defined security-related fields at data ingestion time and storing the extracted data in a relational database. This traditional data extraction process (and associated reduction in data size) that occurs at data ingestion time inevitably hampers future incident investigations that may need original data to determine the root cause of a security issue, or to detect the onset of an impending security threat.

In contrast, the SPLUNK® APP FOR ENTERPRISE SECURITY system stores large volumes of minimally processed security-related data at ingestion time for later retrieval and analysis at search time when a live security threat is being investigated. To facilitate this data retrieval process, the SPLUNK® APP FOR ENTERPRISE SECURITY provides pre-specified schemas for extracting relevant values from the different types of security-related event data and enables a user to define such schemas.

The SPLUNK® APP FOR ENTERPRISE SECURITY can process many types of security-related information. In general, this security-related information can include any information that can be used to identify security threats. For example, the security-related information can include network-related information, such as IP addresses, domain names, asset identifiers, network traffic volume, uniform resource locator strings, and source addresses. The process of detecting security threats for network-related information is further described in U.S. Pat. No. 8,826,434, entitled "SECURITY THREAT DETECTION BASED ON INDICATIONS IN BIG DATA OF ACCESS TO NEWLY REGISTERED DOMAINS", issued on 2 Sep. 2014, U.S. patent application Ser. No. 13/956,252, entitled "INVESTIGATIVE AND DYNAMIC DETECTION OF POTENTIAL SECURITY-THREAT INDICATORS FROM EVENTS IN BIG DATA", filed on 31 Jul. 2013, U.S. patent application Ser. No. 14/445,018, entitled "GRAPHIC DISPLAY OF SECURITY THREATS BASED ON INDICATIONS OF ACCESS TO NEWLY REGISTERED DOMAINS", filed on 28 Jul. 2014, U.S. patent application Ser. No. 14/445,023, entitled "SECURITY THREAT DETECTION OF NEWLY REGISTERED DOMAINS", filed on 28 Jul. 2014, U.S. patent application Ser. No. 14/815,971, entitled "SECURITY THREAT DETECTION USING DOMAIN NAME ACCESSES", filed on 1 Aug. 2015, and U.S. patent application Ser. No. 14/815,972, entitled "SECURITY THREAT DETECTION USING DOMAIN NAME REGISTRATIONS", filed on 1 Aug. 2015, each of which is hereby incorporated by reference in its entirety for all purposes. Security-related information can also include malware infection data and system configuration information, as well as access control information, such as login/logout information and access failure notifications. The security-related information can originate from various sources within a data center, such as hosts, virtual machines, storage devices and sensors. The security-related information can also originate from various sources in a network, such as routers, switches, email servers, proxy servers, gateways, firewalls and intrusion-detection systems.

During operation, the SPLUNK® APP FOR ENTERPRISE SECURITY facilitates detecting "notable events" that are likely to indicate a security threat. These notable events can be detected in a number of ways: (1) a user can notice a correlation in the data and can manually identify a corresponding group of one or more events as "notable;" or (2) a user can define a "correlation search" specifying criteria for a notable event, and every time one or more events satisfy the criteria, the application can indicate that the one or more events are notable. A user can alternatively select a pre-defined correlation search provided by the application. Note that correlation searches can be run continuously or at regular intervals (e.g., every hour) to search for notable events. Upon detection, notable events can be stored in a dedicated "notable events index," which can be subsequently accessed to generate various visualizations containing security-related information. Also, alerts can be generated to notify system operators when important notable events are discovered.

The SPLUNK® APP FOR ENTERPRISE SECURITY provides various visualizations to aid in discovering security threats, such as a "key indicators view" that enables a user to view security metrics, such as counts of different types of notable events. For example, FIG. **9**A illustrates an example key indicators view **900** that comprises a dashboard, which can display a value **901**, for various security-related metrics, such as malware infections **902**. It can also display a change in a metric value **903**, which indicates that the number of malware infections increased by 63 during the preceding interval. Key indicators view **900** additionally displays a histogram panel **904** that displays a histogram of notable events organized by urgency values, and a histogram of notable events organized by time intervals. This key indicators view is described in further detail in pending U.S. patent application Ser. No. 13/956,338, entitled "KEY INDICATORS VIEW", filed on 31 Jul. 2013, and which is hereby incorporated by reference in its entirety for all purposes.

These visualizations can also include an "incident review dashboard" that enables a user to view and act on "notable events." These notable events can include: (1) a single event of high importance, such as any activity from a known web attacker; or (2) multiple events that collectively warrant review, such as a large number of authentication failures on a host followed by a successful authentication. For example, FIG. **9**B illustrates an example incident review dashboard **910** that includes a set of incident attribute fields **911** that, for example, enables a user to specify a time range field **912** for the displayed events. It also includes a timeline **913** that graphically illustrates the number of incidents that occurred in time intervals over the selected time range. It additionally displays an events list **914** that enables a user to view a list of all of the notable events that match the criteria in the incident attributes fields **911**. To facilitate identifying patterns among the notable events, each notable event can be associated with an urgency value (e.g., low, medium, high, critical), which is indicated in the incident review dashboard. The urgency value for a detected event can be determined based on the severity of the event and the priority of the system component associated with the event.

2.12. Data Center Monitoring

As mentioned above, the SPLUNK® ENTERPRISE platform provides various features that simplify the developers's task to create various applications. One such application is SPLUNK® APP FOR VMWARE® that provides operational visibility into granular performance metrics, logs, tasks and events, and topology from hosts, virtual machines and virtual centers. It empowers administrators with an accurate real-time picture of the health of the environment, proactively identifying performance and capacity bottlenecks.

Conventional data-center-monitoring systems lack the infrastructure to effectively store and analyze large volumes of machine-generated data, such as performance information and log data obtained from the data center. In conventional data-center-monitoring systems, machine-generated data is typically pre-processed prior to being stored, for example, by extracting pre-specified data items and storing them in a database to facilitate subsequent retrieval and analysis at search time. However, the rest of the data is not saved and discarded during pre-processing.

In contrast, the SPLUNK® APP FOR VMWARE® stores large volumes of minimally processed machine data, such as performance information and log data, at ingestion time for later retrieval and analysis at search time when a live performance issue is being investigated. In addition to data obtained from various log files, this performance-related information can include values for performance metrics obtained through an application programming interface (API) provided as part of the vSphere Hypervisor™ system distributed by VMware, Inc. of Palo Alto, Calif. For example, these performance metrics can include: (1) CPU-related performance metrics; (2) disk-related performance metrics; (3) memory-related performance metrics; (4) network-related performance metrics; (5) energy-usage statistics; (6) data-traffic-related performance metrics; (7) overall system availability performance metrics; (8) cluster-related performance metrics; and (9) virtual machine performance statistics. Such performance metrics are described in U.S. patent application Ser. No. 14/167,316, entitled "CORRELATION FOR USER-SELECTED TIME RANGES OF VALUES FOR PERFORMANCE METRICS OF COMPONENTS IN AN INFORMATION-TECHNOLOGY ENVIRONMENT WITH LOG DATA FROM THAT INFORMATION-TECHNOLOGY ENVIRONMENT", filed on 29 Jan. 2014, and which is hereby incorporated by reference in its entirety for all purposes.

To facilitate retrieving information of interest from performance data and log files, the SPLUNK® APP FOR VMWARE® provides pre-specified schemas for extracting relevant values from different types of performance-related event data, and also enables a user to define such schemas.

The SPLUNK® APP FOR VMWARE® additionally provides various visualizations to facilitate detecting and diagnosing the root cause of performance problems. For example, one such visualization is a "proactive monitoring tree" that enables a user to easily view and understand relationships among various factors that affect the performance of a hierarchically structured computing system. This proactive monitoring tree enables a user to easily navigate the hierarchy by selectively expanding nodes representing various entities (e.g., virtual centers or computing clusters) to view performance information for lower-level nodes associated with lower-level entities (e.g., virtual machines or host systems). Example node-expansion operations are illustrated in FIG. **9**C, wherein nodes **933** and **934** are selectively expanded. Note that nodes **931-939** can be displayed using different patterns or colors to represent different performance states, such as a critical state, a warning state, a normal state or an unknown/offline state. The ease of navigation provided by selective expansion in combination with the associated performance-state information enables a user to quickly diagnose the root cause of a performance problem. The proactive monitoring tree is described in further detail in U.S. patent application Ser. No. 14/253,490, entitled "PROACTIVE MONITORING TREE WITH SEVERITY STATE SORTING", filed on 15 Apr. 2014, and U.S. patent application Ser. No. 14/812,948, also entitled "PROACTIVE MONITORING TREE WITH SEVERITY STATE SORTING", filed on 29 Jul. 2015, each of which is hereby incorporated by reference in its entirety for all purposes.

The SPLUNK® APP FOR VMWARE® also provides a user interface that enables a user to select a specific time range and then view heterogeneous data comprising events, log data, and associated performance metrics for the selected time range. For example, the screen illustrated in FIG. **9**D displays a listing of recent "tasks and events" and a listing of recent "log entries" for a selected time range above a performance-metric graph for "average CPU core utilization" for the selected time range. Note that a user is able to operate pull-down menus **942** to selectively display different performance metric graphs for the selected time range. This enables the user to correlate trends in the performance-metric graph with corresponding event and log data to quickly determine the root cause of a performance problem. This user interface is described in more detail in U.S. patent application Ser. No. 14/167,316, entitled "CORRELATION FOR USER-SELECTED TIME RANGES OF VALUES FOR PERFORMANCE METRICS OF COMPONENTS IN AN INFORMATION-TECHNOLOGY ENVIRONMENT WITH LOG DATA FROM THAT INFORMATION-TECHNOLOGY ENVIRONMENT", filed on 29 Jan. 2014, and which is hereby incorporated by reference in its entirety for all purposes.

2.13. Cloud-Based System Overview

The example data intake and query system **108** described in reference to FIG. **2** comprises several system components, including one or more forwarders, indexers, and search heads. In some environments, a user of a data intake and query system **108** may install and configure, on computing devices owned and operated by the user, one or more software applications that implement some or all of these system components. For example, a user may install a software application on server computers owned by the user and configure each server to operate as one or more of a forwarder, an indexer, a search head, etc. This arrangement generally may be referred to as an "on-premises" solution. That is, the system **108** is installed and operates on computing devices directly controlled by the user of the system. Some users may prefer an on-premises solution because it may provide a greater level of control over the configuration of certain aspects of the system (e.g., security, privacy, standards, controls, etc.). However, other users may instead prefer an arrangement in which the user is not directly responsible for providing and managing the computing devices upon which various components of system **108** operate.

In one embodiment, to provide an alternative to an entirely on-premises environment for system **108**, one or more of the components of a data intake and query system instead may be provided as a cloud-based service. In this context, a cloud-based service refers to a service hosted by one more computing resources that are accessible to end

users over a network, for example, by using a web browser or other application on a client device to interface with the remote computing resources. For example, a service provider may provide a cloud-based data intake and query system by managing computing resources configured to implement various aspects of the system (e.g., forwarders, indexers, search heads, etc.) and by providing access to the system to end users via a network. Typically, a user may pay a subscription or other fee to use such a service. Each subscribing user of the cloud-based service may be provided with an account that enables the user to configure a customized cloud-based system based on the user's preferences.

FIG. **10** illustrates a block diagram of an example cloud-based data intake and query system. Similar to the system of FIG. **2**, the networked computer system **1000** includes input data sources **202** and forwarders **204**. These input data sources and forwarders may be in a subscriber's private computing environment. Alternatively, they might be directly managed by the service provider as part of the cloud service. In the example system **1000**, one or more forwarders **204** and client devices **1002** are coupled to a cloud-based data intake and query system **1006** via one or more networks **1004**. Network **1004** broadly represents one or more LANs, WANs, cellular networks, intranetworks, internetworks, etc., using any of wired, wireless, terrestrial microwave, satellite links, etc., and may include the public Internet, and is used by client devices **1002** and forwarders **204** to access the system **1006**. Similar to the system of **108**, each of the forwarders **204** may be configured to receive data from an input source and to forward the data to other components of the system **1006** for further processing.

In an embodiment, a cloud-based data intake and query system **1006** may comprise a plurality of system instances **1008**. In general, each system instance **1008** may include one or more computing resources managed by a provider of the cloud-based system **1006** made available to a particular subscriber. The computing resources comprising a system instance **1008** may, for example, include one or more servers or other devices configured to implement one or more forwarders, indexers, search heads, and other components of a data intake and query system, similar to system **108**. As indicated above, a subscriber may use a web browser or other application of a client device **1002** to access a web portal or other interface that enables the subscriber to configure an instance **1008**.

Providing a data intake and query system as described in reference to system **108** as a cloud-based service presents a number of challenges. Each of the components of a system **108** (e.g., forwarders, indexers and search heads) may at times refer to various configuration files stored locally at each component. These configuration files typically may involve some level of user configuration to accommodate particular types of data a user desires to analyze and to account for other user preferences. However, in a cloud-based service context, users typically may not have direct access to the underlying computing resources implementing the various system components (e.g., the computing resources comprising each system instance **1008**) and may desire to make such configurations indirectly, for example, using one or more web-based interfaces. Thus, the techniques and systems described herein for providing user interfaces that enable a user to configure source type definitions are applicable to both on-premises and cloud-based service contexts, or some combination thereof (e.g., a hybrid system where both an on-premises environment such as SPLUNK® ENTERPRISE and a cloud-based environment such as SPLUNK CLOUD™ are centrally visible).

2.14. Searching Externally Archived Data

FIG. **11** shows a block diagram of an example of a data intake and query system **108** that provides transparent search facilities for data systems that are external to the data intake and query system. Such facilities are available in the HUNK® system provided by Splunk Inc. of San Francisco, Calif. HUNK® represents an analytics platform that enables business and IT teams to rapidly explore, analyze, and visualize data in Hadoop and NoSQL data stores.

The search head **210** of the data intake and query system receives search requests from one or more client devices **1104** over network connections **1120**. As discussed above, the data intake and query system **108** may reside in an enterprise location, in the cloud, etc. FIG. **11** illustrates that multiple client devices **1104***a*, **1104***b*, . . . , **1104***n* may communicate with the data intake and query system **108**. The client devices **1104** may communicate with the data intake and query system using a variety of connections. For example, one client device in FIG. **11** is illustrated as communicating over an Internet (Web) protocol, another client device is illustrated as communicating via a command line interface, and another client device is illustrated as communicating via a system developer kit (SDK).

The search head **210** analyzes the received search request to identify request parameters. If a search request received from one of the client devices **1104** references an index maintained by the data intake and query system, then the search head **210** connects to one or more indexers **206** of the data intake and query system for the index referenced in the request parameters. That is, if the request parameters of the search request reference an index, then the search head accesses the data in the index via the indexer. The data intake and query system **108** may include one or more indexers **206**, depending on system access resources and requirements. As described further below, the indexers **206** retrieve data from their respective local data stores **208** as specified in the search request. The indexers and their respective data stores can comprise one or more storage devices and typically reside on the same system, though they may be connected via a local network connection.

If the request parameters of the received search request reference an external data collection, which is not accessible to the indexers **206** or under the management of the data intake and query system, then the search head **210** can access the external data collection through an External Result Provider (ERP) process **1110**. An external data collection may be referred to as a "virtual index" (plural, "virtual indices"). An ERP process provides an interface through which the search head **210** may access virtual indices.

Thus, a search reference to an index of the system relates to a locally stored and managed data collection. In contrast, a search reference to a virtual index relates to an externally stored and managed data collection, which the search head may access through one or more ERP processes **1110**, **1112**. FIG. **11** shows two ERP processes **1110**, **1112** that connect to respective remote (external) virtual indices, which are indicated as a Hadoop or another system **1114** (e.g., Amazon S3, Amazon EMR, other Hadoop Compatible File Systems (HCFS), etc.) and a relational database management system (RDBMS) **1116**. Other virtual indices may include other file organizations and protocols, such as Structured Query Language (SQL) and the like. The ellipses between the ERP processes **1110**, **1112** indicate optional additional ERP processes of the data intake and query system **108**. An ERP process may be a computer process that is initiated or spawned by the search head **210** and is executed by the

search data intake and query system **108**. Alternatively or additionally, an ERP process may be a process spawned by the search head **210** on the same or different host system as the search head **210** resides.

The search head **210** may spawn a single ERP process in response to multiple virtual indices referenced in a search request, or the search head may spawn different ERP processes for different virtual indices. Generally, virtual indices that share common data configurations or protocols may share ERP processes. For example, all search query references to a Hadoop file system may be processed by the same ERP process, if the ERP process is suitably configured. Likewise, all search query references to an SQL database may be processed by the same ERP process. In addition, the search head may provide a common ERP process for common external data source types (e.g., a common vendor may utilize a common ERP process, even if the vendor includes different data storage system types, such as Hadoop and SQL). Common indexing schemes also may be handled by common ERP processes, such as flat text files or Weblog files.

The search head **210** determines the number of ERP processes to be initiated via the use of configuration parameters that are included in a search request message. Generally, there is a one-to-many relationship between an external results provider "family" and ERP processes. There is also a one-to-many relationship between an ERP process and corresponding virtual indices that are referred to in a search request. For example, using RDBMS, assume two independent instances of such a system by one vendor, such as one RDBMS for production and another RDBMS used for development. In such a situation, it is likely preferable (but optional) to use two ERP processes to maintain the independent operation as between production and development data. Both of the ERPs, however, will belong to the same family, because the two RDBMS system types are from the same vendor.

The ERP processes **1110**, **1112** receive a search request from the search head **210**. The search head may optimize the received search request for execution at the respective external virtual index. Alternatively, the ERP process may receive a search request as a result of analysis performed by the search head or by a different system process. The ERP processes **1110**, **1112** can communicate with the search head **210** via conventional input/output routines (e.g., standard in/standard out, etc.). In this way, the ERP process receives the search request from a client device such that the search request may be efficiently executed at the corresponding external virtual index.

The ERP processes **1110**, **1112** may be implemented as a process of the data intake and query system. Each ERP process may be provided by the data intake and query system, or may be provided by process or application providers who are independent of the data intake and query system. Each respective ERP process may include an interface application installed at a computer of the external result provider that ensures proper communication between the search support system and the external result provider. The ERP processes **1110**, **1112** generate appropriate search requests in the protocol and syntax of the respective virtual indices **1114**, **1116**, each of which corresponds to the search request received by the search head **210**. Upon receiving search results from their corresponding virtual indices, the respective ERP process passes the result to the search head **210**, which may return or display the results or a processed set of results based on the returned results to the respective client device.

Client devices **1104** may communicate with the data intake and query system **108** through a network interface **1120**, e.g., one or more LANs, WANs, cellular networks, intranetworks, and/or internetworks using any of wired, wireless, terrestrial microwave, satellite links, etc., and may include the public Internet.

The analytics platform utilizing the External Result Provider process described in more detail in U.S. Pat. No. 8,738,629, entitled "EXTERNAL RESULT PROVIDED PROCESS FOR RETRIEVING DATA STORED USING A DIFFERENT CONFIGURATION OR PROTOCOL", issued on 27 May 2014, U.S. Pat. No. 8,738,587, entitled "PROCESSING A SYSTEM SEARCH REQUEST BY RETRIEVING RESULTS FROM BOTH A NATIVE INDEX AND A VIRTUAL INDEX", issued on 25 Jul. 2013, U.S. patent application Ser. No. 14/266,832, entitled "PROCESSING A SYSTEM SEARCH REQUEST ACROSS DISPARATE DATA COLLECTION SYSTEMS", filed on 1 May 2014, and U.S. patent application Ser. No. 14/449,144, entitled "PROCESSING A SYSTEM SEARCH REQUEST INCLUDING EXTERNAL DATA SOURCES", filed on 31 Jul. 2014, each of which is hereby incorporated by reference in its entirety for all purposes.

2.14.1. ERP Process Features

The ERP processes described above may include two operation modes: a streaming mode and a reporting mode. The ERP processes can operate in streaming mode only, in reporting mode only, or in both modes simultaneously. Operating in both modes simultaneously is referred to as mixed mode operation. In a mixed mode operation, the ERP at some point can stop providing the search head with streaming results and only provide reporting results thereafter, or the search head at some point may start ignoring streaming results it has been using and only use reporting results thereafter.

The streaming mode returns search results in real time, with minimal processing, in response to the search request. The reporting mode provides results of a search request with processing of the search results prior to providing them to the requesting search head, which in turn provides results to the requesting client device. ERP operation with such multiple modes provides greater performance flexibility with regard to report time, search latency, and resource utilization.

In a mixed mode operation, both streaming mode and reporting mode are operating simultaneously. The streaming mode results (e.g., the raw data obtained from the external data source) are provided to the search head, which can then process the results data (e.g., break the raw data into events, timestamp it, filter it, etc.) and integrate the results data with the results data from other external data sources, and/or from data stores of the search head. The search head performs such processing and can immediately start returning interim (streaming mode) results to the user at the requesting client device; simultaneously, the search head is waiting for the ERP process to process the data it is retrieving from the external data source as a result of the concurrently executing reporting mode.

In some instances, the ERP process initially operates in a mixed mode, such that the streaming mode operates to enable the ERP quickly to return interim results (e.g., some of the raw or unprocessed data necessary to respond to a search request) to the search head, enabling the search head to process the interim results and begin providing to the client or search requester interim results that are responsive to the query. Meanwhile, in this mixed mode, the ERP also operates concurrently in reporting mode, processing por-

tions of raw data in a manner responsive to the search query. Upon determining that it has results from the reporting mode available to return to the search head, the ERP may halt processing in the mixed mode at that time (or some later time) by stopping the return of data in streaming mode to the search head and switching to reporting mode only. The ERP at this point starts sending interim results in reporting mode to the search head, which in turn may then present this processed data responsive to the search request to the client or search requester. Typically the search head switches from using results from the ERP's streaming mode of operation to results from the ERP's reporting mode of operation when the higher bandwidth results from the reporting mode outstrip the amount of data processed by the search head in the]streaming mode of ERP operation.

A reporting mode may have a higher bandwidth because the ERP does not have to spend time transferring data to the search head for processing all the raw data. In addition, the ERP may optionally direct another processor to do the processing.

The streaming mode of operation does not need to be stopped to gain the higher bandwidth benefits of a reporting mode; the search head could simply stop using the streaming mode results—and start using the reporting mode results—when the bandwidth of the reporting mode has caught up with or exceeded the amount of bandwidth provided by the streaming mode. Thus, a variety of triggers and ways to accomplish a search head's switch from using streaming mode results to using reporting mode results may be appreciated by one skilled in the art.

The reporting mode can involve the ERP process (or an external system) performing event breaking, time stamping, filtering of events to match the search query request, and calculating statistics on the results. The user can request particular types of data, such as if the search query itself involves types of events, or the search request may ask for statistics on data, such as on events that meet the search request. In either case, the search head understands the query language used in the received query request, which may be a proprietary language. One exemplary query language is Splunk Processing Language (SPL) developed by the assignee of the application, Splunk Inc. The search head typically understands how to use that language to obtain data from the indexers, which store data in a format used by the SPLUNK® Enterprise system.

The ERP processes support the search head, as the search head is not ordinarily configured to understand the format in which data is stored in external data sources such as Hadoop or SQL data systems. Rather, the ERP process performs that translation from the query submitted in the search support system's native format (e.g., SPL if SPLUNK® ENTERPRISE is used as the search support system) to a search query request format that will be accepted by the corresponding external data system. The external data system typically stores data in a different format from that of the search support system's native index format, and it utilizes a different query language (e.g., SQL or MapReduce, rather than SPL or the like).

As noted, the ERP process can operate in the streaming mode alone. After the ERP process has performed the translation of the query request and received raw results from the streaming mode, the search head can integrate the returned data with any data obtained from local data sources (e.g., native to the search support system), other external data sources, and other ERP processes (if such operations were required to satisfy the terms of the search query). An advantage of mixed mode operation is that, in addition to streaming mode, the ERP process is also executing concurrently in reporting mode. Thus, the ERP process (rather than the search head) is processing query results (e.g., performing event breaking, timestamping, filtering, possibly calculating statistics if required to be responsive to the search query request, etc.). It should be apparent to those skilled in the art that additional time is needed for the ERP process to perform the processing in such a configuration. Therefore, the streaming mode will allow the search head to start returning interim results to the user at the client device before the ERP process can complete sufficient processing to start returning any search results. The switchover between streaming and reporting mode happens when the ERP process determines that the switchover is appropriate, such as when the ERP process determines it can begin returning meaningful results from its reporting mode.

The operation described above illustrates the source of operational latency: streaming mode has low latency (immediate results) and usually has relatively low bandwidth (fewer results can be returned per unit of time). In contrast, the concurrently running reporting mode has relatively high latency (it has to perform a lot more processing before returning any results) and usually has relatively high bandwidth (more results can be processed per unit of time). For example, when the ERP process does begin returning report results, it returns more processed results than in the streaming mode, because, e.g., statistics only need to be calculated to be responsive to the search request. That is, the ERP process doesn't have to take time to first return raw data to the search head. As noted, the ERP process could be configured to operate in streaming mode alone and return just the raw data for the search head to process in a way that is responsive to the search request. Alternatively, the ERP process can be configured to operate in the reporting mode only. Also, the ERP process can be configured to operate in streaming mode and reporting mode concurrently, as described, with the ERP process stopping the transmission of streaming results to the search head when the concurrently running reporting mode has caught up and started providing results. The reporting mode does not require the processing of all raw data that is responsive to the search query request before the ERP process starts returning results; rather, the reporting mode usually performs processing of chunks of events and returns the processing results to the search head for each chunk.

For example, an ERP process can be configured to merely return the contents of a search result file verbatim, with little or no processing of results. That way, the search head performs all processing (such as parsing byte streams into events, filtering, etc.). The ERP process can be configured to perform additional intelligence, such as analyzing the search request and handling all the computation that a native search indexer process would otherwise perform. In this way, the configured ERP process provides greater flexibility in features while operating according to desired preferences, such as response latency and resource requirements.

2.14. IT Service Monitoring

As previously mentioned, the SPLUNK® ENTERPRISE platform provides various schemas, dashboards and visualizations that make it easy for developers to create applications to provide additional capabilities. One such application is SPLUNK® IT SERVICE INTELLIGENCE™, which performs monitoring and alerting operations. It also includes analytics to help an analyst diagnose the root cause of performance problems based on large volumes of data stored by the SPLUNK® ENTERPRISE system as correlated to the various services an IT organization provides (a service-

centric view). This differs significantly from conventional IT monitoring systems that lack the infrastructure to effectively store and analyze large volumes of service-related event data. Traditional service monitoring systems typically use fixed schemas to extract data from pre-defined fields at data ingestion time, wherein the extracted data is typically stored in a relational database. This data extraction process and associated reduction in data content that occurs at data ingestion time inevitably hampers future investigations, when all of the original data may be needed to determine the root cause of or contributing factors to a service issue.

In contrast, a SPLUNK® IT SERVICE INTELLIGENCE™ system stores large volumes of minimally-processed service-related data at ingestion time for later retrieval and analysis at search time, to perform regular monitoring, or to investigate a service issue. To facilitate this data retrieval process, SPLUNK® IT SERVICE INTELLIGENCE™ enables a user to define an IT operations infrastructure from the perspective of the services it provides. In this service-centric approach, a service such as corporate e-mail may be defined in terms of the entities employed to provide the service, such as host machines and network devices. Each entity is defined to include information for identifying all of the event data that pertains to the entity, whether produced by the entity itself or by another machine, and considering the many various ways the entity may be identified in raw machine data (such as by a URL, an IP address, or machine name). The service and entity definitions can organize event data around a service so that all of the event data pertaining to that service can be easily identified. This capability provides a foundation for the implementation of Key Performance Indicators.

One or more Key Performance Indicators (KPI's) are defined for a service within the SPLUNK® IT SERVICE INTELLIGENCE™ application. Each KPI measures an aspect of service performance at a point in time or over a period of time (aspect KPI's). Each KPI is defined by a search query that derives a KPI value from the machine data of events associated with the entities that provide the service. Information in the entity definitions may be used to identify the appropriate events at the time a KPI is defined or whenever a KPI value is being determined. The KPI values derived over time may be stored to build a valuable repository of current and historical performance information for the service, and the repository, itself, may be subject to search query processing. Aggregate KPIs may be defined to provide a measure of service performance calculated from a set of service aspect KPI values; this aggregate may even be taken across defined timeframes and/or across multiple services. A particular service may have an aggregate KPI derived from substantially all of the aspect KPI's of the service to indicate an overall health score for the service.

SPLUNK® IT SERVICE INTELLIGENCE™ facilitates the production of meaningful aggregate KPI's through a system of KPI thresholds and state values. Different KPI definitions may produce values in different ranges, and so the same value may mean something very different from one KPI definition to another. To address this, SPLUNK® IT SERVICE INTELLIGENCE™ implements a translation of individual KPI values to a common domain of "state" values. For example, a KPI range of values may be 1-100, or 50-275, while values in the state domain may be 'critical,' 'warning,' 'normal,' and 'informational'. Thresholds associated with a particular KPI definition determine ranges of values for that KPI that correspond to the various state values. In one case, KPI values 95-100 may be set to correspond to 'critical' in the state domain. KPI values from disparate KPI's can be processed uniformly once they are translated into the common state values using the thresholds. For example, "normal 80% of the time" can be applied across various KPI's. To provide meaningful aggregate KPI's, a weighting value can be assigned to each KPI so that its influence on the calculated aggregate KPI value is increased or decreased relative to the other KPI's.

One service in an IT environment often impacts, or is impacted by, another service. SPLUNK® IT SERVICE INTELLIGENCE™ can reflect these dependencies. For example, a dependency relationship between a corporate e-mail service and a centralized authentication service can be reflected by recording an association between their respective service definitions. The recorded associations establish a service dependency topology that informs the data or selection options presented in a GUI, for example. (The service dependency topology is like a "map" showing how services are connected based on their dependencies.) The service topology may itself be depicted in a GUI and may be interactive to allow navigation among related services.

Entity definitions in SPLUNK® IT SERVICE INTELLIGENCE™ can include informational fields that can serve as metadata, implied data fields, or attributed data fields for the events identified by other aspects of the entity definition. Entity definitions in SPLUNK® IT SERVICE INTELLIGENCE™ can also be created and updated by an import of tabular data (as represented in a CSV, another delimited file, or a search query result set). The import may be GUI-mediated or processed using import parameters from a GUI-based import definition process. Entity definitions in SPLUNK® IT SERVICE INTELLIGENCE™ can also be associated with a service by means of a service definition rule. Processing the rule results in the matching entity definitions being associated with the service definition. The rule can be processed at creation time, and thereafter on a scheduled or on-demand basis. This allows dynamic, rule-based updates to the service definition.

During operation, SPLUNK® IT SERVICE INTELLIGENCE™ can recognize so-called "notable events" that may indicate a service performance problem or other situation of interest. These notable events can be recognized by a "correlation search" specifying trigger criteria for a notable event: every time KPI values satisfy the criteria, the application indicates a notable event. A severity level for the notable event may also be specified. Furthermore, when trigger criteria are satisfied, the correlation search may additionally or alternatively cause a service ticket to be created in an IT service management (ITSM) system, such as a systems available from ServiceNow, Inc., of Santa Clara, Calif.

SPLUNK® IT SERVICE INTELLIGENCE™ provides various visualizations built on its service-centric organization of event data and the KPI values generated and collected. Visualizations can be particularly useful for monitoring or investigating service performance. SPLUNK® IT SERVICE INTELLIGENCE™ provides a service monitoring interface suitable as the home page for ongoing IT service monitoring. The interface is appropriate for settings such as desktop use or for a wall-mounted display in a network operations center (NOC). The interface may prominently display a services health section with tiles for the aggregate KPI's indicating overall health for defined services and a general KPI section with tiles for KPI's related to individual service aspects. These tiles may display KPI information in a variety of ways, such as by being colored and ordered according to factors like the KPI state value.

They also can be interactive and navigate to visualizations of more detailed KPI information.

SPLUNK® IT SERVICE INTELLIGENCE™ provides a service-monitoring dashboard visualization based on a user-defined template. The template can include user-selectable widgets of varying types and styles to display KPI information. The content and the appearance of widgets can respond dynamically to changing KPI information. The KPI widgets can appear in conjunction with a background image, user drawing objects, or other visual elements, that depict the IT operations environment, for example. The KPI widgets or other GUI elements can be interactive so as to provide navigation to visualizations of more detailed KPI information.

SPLUNK® IT SERVICE INTELLIGENCE™ provides a visualization showing detailed time-series information for multiple KPI's in parallel graph lanes. The length of each lane can correspond to a uniform time range, while the width of each lane may be automatically adjusted to fit the displayed KPI data. Data within each lane may be displayed in a user selectable style, such as a line, area, or bar chart. During operation a user may select a position in the time range of the graph lanes to activate lane inspection at that point in time. Lane inspection may display an indicator for the selected time across the graph lanes and display the KPI value associated with that point in time for each of the graph lanes. The visualization may also provide navigation to an interface for defining a correlation search, using information from the visualization to pre-populate the definition.

SPLUNK® IT SERVICE INTELLIGENCE™ provides a visualization for incident review showing detailed information for notable events. The incident review visualization may also show summary information for the notable events over a time frame, such as an indication of the number of notable events at each of a number of severity levels. The severity level display may be presented as a rainbow chart with the warmest color associated with the highest severity classification. The incident review visualization may also show summary information for the notable events over a time frame, such as the number of notable events occurring within segments of the time frame. The incident review visualization may display a list of notable events within the time frame ordered by any number of factors, such as time or severity. The selection of a particular notable event from the list may display detailed information about that notable event, including an identification of the correlation search that generated the notable event.

SPLUNK® IT SERVICE INTELLIGENCE™ provides pre-specified schemas for extracting relevant values from the different types of service-related event data. It also enables a user to define such schemas.

3.0 Overview of Enhancing Time Series Predictions

Data is often collected as a time series data set, that is, a sequence of data points, typically including successive measurements made over a time interval. Time series data is frequently utilized to perform a predictive or forecasting data analysis. In this manner, time series data can be used to predict a future outcome based on the historical time series data. Expected outcomes can be predicted in relation to any type or number of data. By way of example only, and without limitation, a user may desire forecasted data values for data center capacity, sales, license uses, etc.

Algorithms for time series forecasting can have many forms. For example, to perform time series forecasting, various algorithms using Kalman filtering can be implemented to forecast or predict expected values using collected time series data. Generally, Kalman filtering refers to an algorithm that uses a series of measurements observed over time and produces estimates of unknown variables. A Kalman filter model includes parameters that are adjusted or calculated using historical time series data. Upon determining the parameters, the generated model can be used to calculate future predictions. Any number or type of prediction models can be generated based on Kalman filtering. For example, a local level algorithm, a seasonal local level, a local level trend, a bivariate local level, or an algorithm combination can be based on the Kalman filter. At a high level, a local level algorithm refers to an algorithm having a univariate model with no or limited trends and seasonality. A seasonal local level algorithm refers to an algorithm having a univariate model with seasonality (periodicity of the time series is computed). A local level trend algorithm refers to an algorithm having a univariate model with trend but no or minimal seasonality. A local level bivariate algorithm refers to a bivariate model with no or limited trends and seasonality. Such an algorithm can use one set of data to make predictions for another. For example, assume the bivariate model uses dataset Y to make predictions for dataset X. If the holdback equals ten, the model takes the last ten data points of Y to make predictions for the last ten data points of X. A combination algorithm can be any combination of the above or additional algorithms. For example, in one embodiment, a combination algorithm may combine a seasonal local level algorithm and a local level trend algorithm to generate the combined algorithm. This combined algorithm benefits from utilizing the periodicity in data of the seasonal local level algorithm and the trend in data of the local level trend algorithm. Although not described herein, other combinations of algorithms can be utilized. Such prediction models can then be used predict data associated with any number of future time intervals.

Typically, forecasting or prediction models generate more accurate predictions when provided with more time series data points. In other words, the more historical data applied or analyzed, the more accurate the prediction. In many cases, however, time series data is missing. Missing time series data can result in less accurate predictions or forecasts. As such, missing data might be predicted such that values for the missing data can be used along with the previously recorded time series data to predict or forecast expected values.

Generally, predict commands are used to perform future predictions for time series data. As such, a predict command initiates prediction or forecasting of expected values based on previously recorded time series data. In addition to predicting a future value, a predict command can also generate an upper and lower boundary or confidence level that indicates a level of accuracy for the predicted value. For instance, the predict command can predict upper and lower confidence intervals for each predicted missing value that specify, for example, where 95% of the predictions are expected to fall. As described herein, the predict command can be used to fill in or predict missing data in a time series. Both the observed time series data and the predicted missing values can be used during the forecasting process. For example, predicted missing values can be used to compute periodicity in association with a time series data set. Periodicity generally refers to a tendency of data to recur at intervals. As such, to compute periodicity, data values, including predicted missing values, can be analyzed to determine a period over which the values recur at intervals, or values repeat in irregular intervals or periods. Predicted missing values can also be used, along with captured time series data, to predict or forecast future values.

By way of example, and with reference to FIGS. **18**A and **18**B, FIG. **18**A illustrates a captured time series data set **1802**. In accordance with entering a predict command **1804**, predicted missing values **1806** in FIG. **18**B are calculated and presented to visually illustrate the predicted missing values that occur between the existing recorded time series data set. As shown in FIG. **18**A, the observed time series data set is delineated with gaps or voids where missing values exist in the time series data, whereas the predicted missing values are included in the visual pattern illustrated in FIG. **18**B. As discussed herein, such predicted missing values may be used to compute periodicity. Upon determining periodicity, historical time series data and predicted missing values can then be used to forecast or predict future values **1808** in FIG. **18**B. As will be described herein, the predicted missing values to utilize for determining periodicity may be the same or different as predicted missing values to forecast expected values. In this regard, one method for predicting missing values may be employed for use in determining periodicity, while another method for predicting missing values may be employed for use in forecasting expected values.

Further, embodiments described herein facilitate concurrent forecasting of multiple time series data sets. In particular, in response to receiving a predict command indicating multiple time series data sets and, optionally, corresponding forecasting algorithms (and other parameters), the predict command can be parsed to identify each of the time series data sets. Based on the specified time series data sets and corresponding parameters, an object is initiated for each specified time series data set to separately, but concurrently, perform predictive analysis. The results from the execution of the separate predictive analysis can be aggregated and provided to a requesting device for display to a user such that the user can simultaneously view the forecasted results for the multiple time series data sets. As can be appreciated, initiating concurrent forecasting of multiple time series data sets improves processing time that would otherwise be required to perform subsequent or serial predictive analysis for each time series data set. In addition, a user can initiate predictive analyses for multiple time series data sets in a single query rather than initiating a predictive analysis for each time series data set independently. In response to the concurrent predictive analyses, the user can be provided with a single view of forecasted results for multiple time series data sets, rather than being separately provided with forecasted results for various time series data sets.

3.1 Overview of a Predictive Analysis Tool in a Data Processing Environment

FIG. **19** illustrates an example data processing environment **1900** in accordance with various embodiments of the present disclosure. Generally, the data processing environment **1900** refers to an environment that provides for, or enables, the management, storage, and retrieval of data to predict or forecast values expected to occur or result in the future. As shown in FIG. **19**, the data processing environment includes a predictive analysis tool **1916** used to predict or forecast expected outcomes. The predictive analysis tool **1916** can utilize historical time series data to generate predicted outcomes expected in the future. As described herein, in addition to using historical time series data, the predictive analysis tool **1916** uses predicted missing values in generating expected or forecasted values. Predicted missing values refer to values that are predicted for data missing in the historical time series data set. As such, predicted missing values are generally values that fall at points in time between recorded or observed values in a time series data set.

In some embodiments, the environment **1900** can include an event-processing system **1902** communicatively coupled to one or more client devices **1904** and one or more data sources **1906** via a communications network **1908**. The network **1908** may include an element or system that facilitates communication between the entities of the environment **1900**. The network **1908** may include an electronic communications network, such as the Internet, a local area network (LAN), a wide area network (WAN), a wireless local area network (WLAN), a cellular communications network, and/or the like. In some embodiments, the network **1908** can include a wired or a wireless network. In some embodiments, the network **1908** can include a single network or a combination of networks.

The data source **1906** may be a source of incoming source data **1910** being fed into the event-processing system **1902**. A data source **1906** can be or include one or more external data sources, such as web servers, application servers, databases, firewalls, routers, operating systems, and software applications that execute on computer systems, mobile devices, sensors, and/or the like. Data source **1906** may be located remote from the event-processing system **1902**. For example, a data source **1906** may be defined on an agent computer operating remote from the event-processing system **1902**, such as on-site at a customer's location, that transmits source data **1910** to event-processing system **1902** via a communications network (e.g., network **1908**).

Source data **1910** can be a stream or set of data fed to an entity of the event-processing system **1902**, such as a forwarder (not shown) or an indexer **1912**. In some embodiments, the source data **1910** can be heterogeneous machine-generated data received from various data sources **1906**, such as servers, databases, applications, networks, and/or the like. Source data **1910** may include, for example raw data (e.g., raw time-series data), such as server log files, activity log files, configuration files, messages, network packet data, performance measurements, sensor measurements, and/or the like. For example, source data **1910** may include log data generated by a server during the normal course of operation (e.g. server log data). In some embodiments, the source data **1910** may be minimally processed to generate minimally processed source data. For example, the source data **1910** may be received from a data source **1906**, such as a server. The source data **1910** may then be subjected to a small amount of processing to break the data into events. As discussed, an event generally refers to a portion, or a segment of the data, that is associated with a time. And, the resulting events may be indexed (e.g., stored in a raw data file associated with an index file). In some embodiments, indexing the source data **1910** may include additional processing, such as compression, replication, and/or the like.

As can be appreciated, source data **1910** might be structured data or unstructured data. Structured data has a predefined format, wherein specific data items with specific data formats reside at predefined locations in the data. For example, data contained in relational databases and spreadsheets may be structured data sets. In contrast, unstructured data does not have a predefined format. This means that unstructured data can comprise various data items having different data types that can reside at different locations.

The indexer **1912** of the event-processing system **1902** receives the source data **1910**, for example, from a forwarder (not shown) or the data source **1906**, and apportions the source data **1910** into events. An indexer **1912** may be an

entity of the event-processing system **1902** that indexes data, transforming source data **1910** into events and placing the results into a data store **1914**, or index. Indexer **1912** may also search data stores **1914** in response to requests or queries. An indexer **1912** may perform other functions, such as data input and search management. In some cases, forwarders (not shown) handle data input, and forward the source data **1910** to the indexers **1912** for indexing.

During indexing, and at a high-level, the indexer **1912** can facilitate taking data from its origin in sources, such as log files and network feeds, to its transformation into searchable events that encapsulate valuable knowledge. The indexer **1912** may acquire a raw data stream (e.g., source data **1910**) from its source (e.g., data source **1906**), break it into blocks (e.g., **64**K blocks of data), and/or annotate each block with metadata keys. After the data has been input, the data can be parsed. This can include, for example, identifying event boundaries, identifying event timestamps (or creating them if they don't exist), masking sensitive event data (such as credit card or social security numbers), applying custom metadata to incoming events, and/or the like. Accordingly, the raw data may be data broken into individual events. The parsed data (also referred to as "events") may be written to a data store, such as an index or data store **1914**.

The data store **1914** may include a medium for the storage of data thereon. For example, data store **1914** may include non-transitory computer-readable medium storing data thereon that is accessible by entities of the environment **1900**, such as the corresponding indexer **1912** and the predictive analysis tool **1916**. As can be appreciated, the data store **1914** may store the data (e.g., events) in any manner. In some implementations, the data may include one or more indexes including one or more buckets, and the buckets may include an index file and/or raw data file (e.g., including parsed, time-stamped events). In some embodiments, each data store is managed by a given indexer that stores data to the data store and/or performs searches of the data stored on the data store. Although certain embodiments are described with regard to a single data store **1914** for purposes of illustration, embodiments may include employing multiple data stores **1914**, such as a plurality of distributed data stores **1914**.

As described, events within the data store **1914** may be represented by a data structure that is associated with a certain point in time and includes a portion of raw machine data (e.g., a portion of machine-generated data that has not been manipulated). An event may include, for example, a line of data that includes a time reference (e.g., a timestamp), and one or more other values. In the context of server log data, for example, an event may correspond to a log entry for a client request and include the following values: (a) a time value (e.g., including a value for the data and time of the request, such as a timestamp), and (b) a series of other values including, for example, a page value (e.g., including a value representing the page requested), an IP (Internet Protocol) value (e.g., including a value for representing the client IP address associated with the request), and an HTTP (Hypertext Transfer protocol) code value (e.g., including a value representative of an HTTP status code), and/or the like. That is, each event may be associated with one or more values. Some events may be associated with default values, such as a host value, a source value, a source type value and/or a time value. A default value may be common to some of all events of a set of source data.

In some embodiments, an event can be associated with one or more characteristics that are not represented by the data initially contained in the raw data, such as characteristics of the host, the source, and/or the source type associated with the event. In the context of server log data, for example, if an event corresponds to a log entry received from Server A, the host and the source of the event may be identified as Server A, and the source type may be determined to be "server." In some embodiments, values representative of the characteristics may be added to (or otherwise associated with) the event. In the context of server log data, for example, if an event is received from Server A, a host value (e.g., including a value representative of Server A), a source value (e.g., including a value representative of Server A), and a source type value (e.g., including a value representative of a "server") may be appended to (or otherwise associated with) the corresponding event.

In some embodiments, events can correspond to data that is generated on a regular basis and/or in response to the occurrence of a given event. In the context of server log data, for example, a server that logs activity every second may generate a log entry every second, and the log entries may be stored as corresponding events of the source data. Similarly, a server that logs data upon the occurrence of an error event may generate a log entry each time an error occurs, and the log entries may be stored as corresponding events of the source data.

In accordance with events being stored in the data store **1914**, the predictive analysis tool **1916** can function to predict or forecast outcomes expected to occur in the future. Although the predictive analysis tool **1916** is illustrated and described herein as a separate component, this is for illustrative purposes. As can be appreciated, the predictive analysis tool **1916**, or functions described in association therewith, can be performed at the indexer **1912**, a search head (not shown), or any other component. For example, some functionality described in association with the predictive analysis tool **1916** might be performed at a search head, while other functionality described in association with the predictive analysis tool **1916** might be performed at an indexer.

As described herein, the predictive analysis tool **1916** can be initiated by a user of the client device **1904**. The client device **1904** may be used or otherwise accessed by a user **1922**, such as a system administrator or a customer. A client device **1904** may include any variety of electronic devices. In some embodiments, a client device **1904** can include a device capable of communicating information via the network **1908**. A client device **1904** may include one or more computer devices, such as a desktop computer, a server, a laptop computer, a tablet computer, a wearable computer device, a personal digital assistant (PDA), a smart phone, and/or the like. In some embodiments, a client device **1904** may be a client of the event processing system **1902**. In some embodiments, a client device **1904** can include various input/output (I/O) interfaces, such as a display (e.g., for displaying a graphical user interface (GUI), an audible output user interface (e.g., a speaker), an audible input user interface (e.g., a microphone), an image acquisition interface (e.g., a camera), a keyboard, a pointer/selection device (e.g., a mouse, a trackball, a touchpad, a touchscreen, a gesture capture or detecting device, or a stylus), and/or the like. In some embodiments, a client device **1904** can include general computing components and/or embedded systems optimized with specific components for performing specific tasks. In some embodiments, a client device **1904** can include programs/applications that can be used to generate a request for content, to provide content, to render content, and/or to send and/or receive requests to and/or from other devices via the network **1908**. For example, a client device **1904** may

include an Internet browser application that facilitates communication with the event-processing system **1902** via the network **1908**. In some embodiments, a program, or application, of a client device **1904** can include program modules having program instructions that are executable by a computer system to perform some or all of the functionality described herein with regard to at least client device **1904**. In some embodiments, a client device **1904** can include one or more computer systems similar to that of the computer system **2600** described below with regard to at least FIG. **26**.

Prediction analysis can be initiated or triggered at the client device **1904** via a search graphical user interface (GUI). In some embodiments, the event-processing system **1902** can provide for the display of a search GUI. Such a search GUI can be displayed on a client device **1904**, and can present information relating to initiating prediction analysis, performing prediction analysis, and viewing results of prediction analysis.

Prediction analysis can be initiated at a client device by a user at any time. In this regard, a user may initiate prediction analysis prior to or in accordance with performing a search for information. Although generally described herein as performing prediction analysis upon the events being created, indexed, and stored, prediction analysis can be defined and applied before or as events are created, indexed, and/or stored. Further, prediction analysis may be automatically triggered. For example, upon initially establishing a prediction analysis, subsequent prediction analyses may be automatically triggered and performed as new data is received.

The prediction analysis tool **1916** is generally configured to facilitate prediction analysis. As described, prediction analysis can be initiated or triggered in response to a user selection or input to predict or forecast data values expected to occur in the future. In this regard, a user may enter a predict command, for example, via client device **1904** to trigger forecasting of future data. An exemplary predict command **1804** is illustrated in FIG. **18**A. In accordance with entering a predict command, or otherwise selecting to initiate prediction analysis, a user can specify additional parameters that can be used to predict or forecast data values. For example, a user might input or select a field name for a variable desired to be predicted, a forecasting algorithm desired for use in predicting outcomes, a timespan or length of time for predicting values into the future (e.g., predict values for the next 6 months), a time interval for predicting values (e.g., predict values for each day), a holdback indicating a number of data points from the end of the time series that are not desired for use in building the forecasting model, a desired confidence level, or the like. Although various parameters might be input by a user, as can be appreciated, in some implementations, any number of such parameters might be default parameters that are used. For example, a default forecasting algorithm might be used or a default confidence level (e.g., 95%) might be used for forecasting data.

Upon the prediction analysis tool **1916** receiving or identifying an indication to initiate prediction analysis, the prediction analysis tool **1916** can access the appropriate time series data for use performing the prediction of expected outcomes. The appropriate time series data to analyze might be indicated, for instance, in the predict command input by a user. As such, the predict command can be parsed to identify various parameters and options related to forecasting, such as, for example, a set of time series data, a prediction time span, an amount of historical data to use for forecasting, a forecasting algorithm to utilize, etc. In embodiments, the time series data can be referenced from the data store **1914**, for example.

As can be appreciated, in some cases, data initially stored in the data store **1914** might not be in a time series data format. For example, raw data stored in the data store **1914** may not be in a time series data format. In such a case, prior to performing prediction analysis to predict or forecast expected values, the data can be converted to a time series data set. One example of converting a set of data into a time series data format includes using a timechart command, or other similar functionality. Generally, the timechart command generates a table of summary statistics. Such data can then be formatted as a chart visualization where the data is plotted against an x-axis that represents a time field. A timechart command may be used to display statistical trends over time, with the option of splitting the data with another field as a separate series in the chart. Timechart visualizations are generally line, area, or column charts.

A timechart command, or other similar functionality, used to convert data into a time series data format can be applied automatically or based on a user selection. For example, upon receiving a predict command, a determination may be made that the corresponding data to use for the predictive analysis is not in a time series data format. As such, a timechart command can be initiated to convert the data into a time series data format. As another example, a user may select or enter a timechart command to convert data into a time series data format such that it is in a format that can be used for data forecasting. Further, conversion of data into a time series data format may occur at any time. For instance, it may occur in association with a selection to perform a predictive analysis, or at a time prior to a selection to perform a predictive analysis. In cases that time series data is not already stored in the data store **1914** for use in predictive analysis, the time series data as determined in accordance with the timechart command can be stored in the data store **1914** for subsequent utilization.

Upon accessing appropriate time series data, the time series data can be analyzed to detect or determine any missing values. Missing values can be detected in any manner and is not intended to be limited to examples provided herein. In an embodiment when input data is converted to a time series data format, such as a table, entries without any values for a particular field can be considered missing in relation to that field. As another example, missing values might be detected when a value is missing in relation to a particular time instance. For example, assume that values are in a time series data set in association with day 1, day 2, and day 4. In such a case, it can be determined that a value is missing in relation to day 3.

For the values determined to be missing, a prediction of the missing value can be made. That is, predictions are generated for missing values (generally referred to herein as predicted missing values). A predicted missing value refers to a predicted or forecasted value associated with a time instance that is missing a collected or observed value. Predicted missing values can be generated based on observed time series data. As such, data previously collected can be used to predict missing values.

In some embodiments, missing values are predicted based on neighboring or nearby time series data. In some cases, a predicted missing value can be determined based on an average, or weighted average, of its neighboring time series data. In this case, the predicted missing values can reflect the trend of the time series data observed before and after the missing data points. One exemplary algorithm used to generate predicted missing values is provided below. In the

below algorithm, assume that a and b represent observed time series data, while the features in between the a and b time series data represent missing values (represented as MV_1, MV_2, MV_3, MV_4). In such a case, the missing values MV_1, MV_2, MV_3, MV_4 can be calculated as follows:

$$\underset{MV_1}{a}, \underset{MV_2}{\frac{na+b}{n+1}}, \underset{MV_3}{\frac{(n-1)a+2b}{n+1}}, \underset{MV_n}{\frac{(n-2)a+3b}{n+1}} \cdots \frac{a+nb}{n+1}, b$$

wherein n denotes the number of missing values between time series data a and b. Such an implementation effectively weights the values based on nearness or proximity to the neighboring value. As a specific example, assume that four missing values are detected MV_1, MV_2, MV_3, MV_4 between a and b time series data. In such a case, the predicted missing values can be determined as:

$$\underset{MV_1}{a}, \underset{MV_2}{\frac{4a+b}{5}}, \underset{MV_3}{\frac{3a+2b}{5}}, \underset{MV_4}{\frac{2a+3b}{5}}, \frac{a+4b}{5}, b$$

As can be appreciated, other algorithms that predict missing values based on neighboring observed time series data are contemplated within the scope of embodiments of the present invention. For example, a predicted missing value can be determined using neighboring time series data in a manner other than using an average, or weighted average, calculation. Further, weights can be applied in other manners to determine the missing values.

Upon generating predicted missing values, the predicted missing values can be used along with the observed time series data to determine periodicity associated with the data. Based on a sequence of values, periodicity is determined to identify a period or cycle associated with the sequence of values. Determining periodicity associated with a set of data, such as observed time series data and predicted missing values, can be performed in any number of ways and is not intended to be limited in scope to method described herein.

In some embodiments, autocorrelation can be used to determine periodicity. Autocorrelation generally refers to cross-correlation of a signal with itself at different points in time. As such, autocorrelation represents the similarity between observations as a function of the time lag between them. Autocorrelation can be used to identify repeating patterns, such as a periodic pattern, for example among a series of values. In implementation, autocorrelations can be determined for any number of lags. A lag generally refers to a period of time or time interval between one event and another (e.g., one data value to another). Autocorrelations of a lag can be generally represented as:

$$\text{Autocorrelation}=a_1a_{1+lag}+a_2a_{2+lag}+\ldots+a_na_n+\text{lag}$$

wherein a_1 represents a first value in a set of data.

In this regard, the autocorrelation of lag 1 is represented as follows:

$$\text{Autocorrelation (lag 1)}=a_1a_2+a_2a_3+a_3a_4+\ldots+a_na_{n+1}+\ldots$$

wherein a_1 represents a first value at a first time in a set of data, a_2 represents a second value at a second time in a set of data, a_3 represents a third value at a third time in a set of data, a_4 represents a fourth value at a fourth time in a set of data, and an represents a nth value at a nth time in a set of data.

Similarly, the autocorrelation of lag 2 is represented as follows:

$$\text{Autocorrelation (lag 2)}=a_1a_3+a_2a_4+a_3a_5+\ldots+a_na_{n+2}+\ldots$$

wherein a_1 represents a first value at a first time in a set of data, a_2 represents a second value at a second time in a set of data, a_3 represents a third value at a third time in a set of data, a_4 represents a fourth value at a fourth time in a set of data, a_5 represents a fifth value at a fifth time in a set of data, and a_n represents a nth value at a nth time in a set of data.

As such, given a set of data, such as observed time series data and predicted missing values, an autocorrelation value can be determined for any number of lags associated with the data. In some embodiments, autocorrelation values are determined for each possible lag value from the first value in the data to the last value in the data set, that is, all lags up to the length of the sequence. In other embodiments, autocorrelation values are determined up to a threshold number of lags. A threshold number can represent the maximum lag value for which autocorrelation is determined. As an example, a threshold number of lag values might equal 2000. In such a case, autocorrelation values are determine for lags 1 to 2000. Such a threshold number (e.g., 2000) may be selected in an effort to maintain performance and accuracy in computing the period. Utilizing a lag threshold can greatly increase computation efficiency, particularly for large data sets. For example, determining autocorrelations of lags from **1** up to the length of the data can result in quadratic time complexity and, as such, decrease the processing speed for a predict command, particularly for large data sets.

Upon determining autocorrelation values for a sequence of data, the autocorrelation values can be searched to identify the greatest or largest autocorrelation value. Stated differently, the autocorrelation values can be analyzed to identify the lag that yields the maximum autocorrelation. In accordance with embodiments of the present invention, the lag associated with the greatest autocorrelation value can be used to designate the periodicity. For example, assume that lag 3 is identified as being associated with the largest autocorrelation value. In such a case, the value of 3 is designated as the periodicity. As another example, assume that the lag 3 is identified as being associated with the largest autocorrelation value. In such a case, the lag plus 1 can be designated as the periodicity (in this case the lag of 3 plus 1 results in a periodicity of 4). As can be appreciated, the periodicity value (e.g., 3 or 4 in these examples) can correspond with any amount of time, such as minutes, hours, days, weeks, etc.

In some cases, prior to designating a lag associated with a largest autocorrelation value as a periodicity for the data, the determined autocorrelation can be compared to a threshold autocorrelation to determine whether to signify the data as being periodic. By way of example only, assume that a threshold autocorrelation of 0.01 is designated. In such a case, if a maximum or greatest correlation is at least 0.01, the corresponding lag value can be returned as a period associated with the data. On the other hand, if the maximum or greatest correlation is equal to or less than 0.01, a value of −1 is returned to indicate that the data is not periodic.

As described, the determined periodicity can be used in association with a forecasting model that considers periodicity to forecast or predict values expected to occur in the future. As can be appreciated, any number of forecasting

models using periodicity can be implemented to provide predictions. Further, in some cases, when forecasting models are combined to predict expected outcomes, one or more of such forecasting models might use periodicity in predicting expected outcomes.

Algorithms for time series forecasting can have many forms. For example, to perform time series forecasting, various algorithms using Kalman filtering can be implemented to forecast or predict expected values using collected time series data. Generally, Kalman filtering refers to an algorithm that uses a series of measurements observed over time and produces estimates of unknown variables. A Kalman filter model includes parameters that are adjusted or calculated using historical time series data. Upon determining the parameters, the generated model can be used to calculate future predictions.

As described, generating parameters for forecasting models can be performed based on historical time series data. Because more data generally results in a more accurate prediction, missing values can be predicted and used along with historical time series data to determine parameters for a forecasting model(s). Missing values can be predicted in any number of ways. For example, in some embodiments, predicted missing values are generated as described above. In this regard, the predicted missing values are determined based on neighbor values. In other embodiments, a different method for determining predicted missing values may be employed. For instance, a Kalman filter recursion can be executed for each missing data point, wherein the output is used to fill in the missing data point.

The resulting predicted missing values and/or forecasted expected values can be provided by the prediction analysis tool **1916** for presentation to the user, for example, via the client device. By way of example, and with reference to FIGS. **18**A and **18**B, FIG. **18**A illustrates a captured time series data set **1802**. In accordance with entering a predict command **1804**, predicted missing values **1806** in FIG. **18**B are calculated and presented to visually illustrate the predicted missing values that occur between the existing recorded time series data set. As shown in FIG. **18**A, the observed time series data set is delineated with gaps or voids where missing values exist in the time series data, whereas the predicted missing values are included in the visual pattern illustrated in FIG. **18**B. As discussed herein, such predicted missing values may be used to compute periodicity and/or to forecast future values. Historical time series data and predicted missing values can then be used to forecast or predict future values **1808** in FIG. **18**B.

Returning to FIG. **19**, in some embodiments, the prediction analysis tool **1916** enables concurrent forecasting in association with multiple time series data sets. In this regard, rather than the performing forecasting for only a single time series data set, forecasting for multiple time series data sets can be performed at the same time.

The prediction analysis tool **1916** can facilitate concurrent prediction analysis of multiple time series data sets. Prediction analysis of multiple time series data sets can be initiated or triggered in response to a user selection or input to concurrently predict or forecast data values expected to occur in the future in association with multiple time series data sets. In this regard, a user may enter a predict command, for example, via client device **1904** to trigger forecasting of future data for multiple time series data sets. An exemplary predict command **2002** is illustrated in FIG. **20**. In accordance with entering a predict command, or otherwise selecting to initiate prediction analysis, a user can specify multiple time series data sets to use for forecasting. Further, additional parameters can be specified that can be used to predict or forecast data values associated with multiple time series data sets. For example, a user might input or select a field name for a variable desired to be predicted, a forecasting algorithm desired for use in predicting outcomes, a timespan or length of time for predicting values into the future (e.g., predict values for the next 6 months), a time interval for predicting values (e.g., predict values for each day), a holdback indicating a number of data points from the end of the time series that are not desired for use in building the forecasting model, a desired confidence level, or the like.

As can be appreciated, in one embodiment, a particular forecasting algorithm can be selected to be applied to each of the desired time series data sets. For instance, a set of five time series data sets may be indicated for performing concurrent prediction analysis with only a single forecasting algorithm set forth. In other embodiments, forecasting algorithms can be selected for the different time series data set. For instance, a different forecasting algorithm can be designated for each specified time series data set. As an example, a set of five time series data sets may be indicated with a different forecasting model specified for each data set.

By way of example only, the exemplary predict command **2002** illustrated in FIG. **20** can initiate concurrent forecasting of values expected in association with two time series data sets, namely "count(Fin)" time series data set **2004** and "Nor" time series data set **2006**. Each indication of the time series data set for forecasting expected values also includes various parameters for performing prediction analysis. For instance, both time series data sets **2004** and **2006** are respectively associated with forecasting algorithms **2008** and **2010**, holdbacks **2012** and **2014**, and timespan **2016** and **2018**. Such an input can initiate concurrent data forecasting in association with the "count (Fin)" time series data set and the "Nor" time series data set using the specified corresponding parameters (e.g., algorithm, etc.).

An indication of time series data sets and associated parameters (e.g., forecasting algorithms) can be specified in any manner. Although FIG. **20** illustrates the time series data sets and associated parameters as being input via a query input box **2020**, as can be appreciated, in some implementations, such selections can be made in other manners. For instance, selection or drop down menus can be employed to make selections of time series data sets, algorithms, or other parameters. Further, although various parameters might be input by a user, as can be appreciated, in some implementations, any number of such parameters might be default parameters that are used. For example, a default forecasting algorithm might be used or a default confidence level (e.g., 95%) might be used for forecasting data.

Upon the prediction analysis tool **1916** receiving or identifying a request to initiate prediction analysis (e.g., via a predict command), the prediction analysis tool **1916** can parse the request to perform concurrent forecasting for multiple time series data sets. A concurrent forecasting request can be parsed to identify specific time series data sets for which forecasting is to be performed. Further, the concurrent forecasting request (e.g., predict command) can be parsed to identify various parameters and options related to forecasting, such as, for example, a prediction time span, an amount of historical data to use for forecasting, a forecasting algorithm to utilize, etc. In some cases, a predict command includes multiple time series data sets with each time series data sets and the corresponding parameter(s) ending before a new time series data set and corresponding parameter(s) begins. When parameters are not specified, a default parameter can be used.

By way of example only, assume that a request to perform concurrent forecasting includes an indication of five time series data sets. In such a case, the concurrent forecasting request can be parsed to identify the five time series data sets. In addition to identifying specific time series data sets for which to perform concurrent forecasting, concurrent forecasting requests can also be parsed to identify other parameters for each time series, such as an algorithm desired to be applied in association with the time series data set, etc.

In accordance with identifying the time series data sets for which concurrent forecasting is desired, the prediction analysis tool **1916** can generate objects to facilitate execution of the specified forecasting algorithms. The number of objects generated generally corresponds with the number of time series data sets such that forecasting algorithms can be separately and independently executed for each time series data set. Stated differently, an object is created for each time series data set. Each object can then independently forecast data in association with the corresponding time series data set. By way of example only, assume concurrent forecasting is to be performed for five different time series data sets. In such a case, five objects are generated or initiated, with each object performing data forecasting for the corresponding time series data set. An object generally refers to a variable(s), a data structure(s), and/or a function(s) (a subroutine is a sequence of program instructions that perform a specific task). In some cases, an object refers to a particular instance of a class where the object can be a combination of variables, functions, and data structures.

In some cases, a parser reads a search query having a predict command and recognizes the multiple time series data sets and corresponding parameters. Software components, or objects, are then constructed for each time series data set. Each object can include the relevant information about a time series data set such as the values and the parameters to be used. Utilizing separate objects can ensure that parameters and/or data for one time series data set do not get mixed up with those associated with another time series data set. Further, using separate objects can ensure that various programming states that arise during processing do not get mixed up with those associated with another time series data set.

As can be appreciated, each object can execute prediction analysis in any manner. For instance, each object might execute prediction analysis as described above. In this manner, the prediction analysis tool **1916** can access the appropriate time series data set for use in performing the prediction of expected outcomes. The appropriate time series data set to analyze might be indicated, for instance, in the predict command input by a user. The time series data set may be referenced and used to generate predicted missing values within the data set. Upon generating predicted missing values, the predicted missing values can be used along with the observed time series data to determine periodicity associated with the data. The determined periodicity can be used in association with the forecasting model associated with the object to forecast or predict values expected to occur in the future. In this regard, the selected forecasting model can be utilized in association with the particular time series data set to forecast data.

Each object or forecasting analysis concurrently performed or executed can then provide forecasted data as output. The forecasted data output for each of the concurrent forecasting analyses can be provided to the client device **1904**. As can be appreciated, in some cases, prior to outputting forecasting data, the data can be combined, aggregated, summarized, or otherwise used to perform calculations to provide results. For instance, upon each object or forecasting analysis being concurrently executed for multiple time series data sets, the prediction analysis tool **1916** can combine the resulting forecasted data and transmit to the requesting client device, such as client device **1904**. In this regard, the forecasted data can be transmitted in association with one another for concurrent display on a display screen.

By way of example only, FIG. **20** provides a user interface illustrating forecasted data associated with concurrent forecasting analyses. The resulting predicted missing values and/or forecasted expected values for multiple time series data sets can be provided by the prediction analysis tool **1916** for presentation to the user, for example, via the client device. FIG. **20** illustrates a captured time series data set for "count(Fin)" **2030** and a captured time series data set for "Nor" **2032**. In accordance with entering the predict command **2002**, predicted missing values **2034** for time series data set **2030** and predicted missing values **2036** for time series data set **2032** are calculated and presented to visually illustrate the predicted missing values that occur between the existing recorded time series data set and/or to predict future values. Further, as illustrated in FIG. **20**, predictions associated with confidence levels can also be included within a forecasting chart or graph. For example, lines **2038** and **2040** represent lower and upper confidence levels (95%) for predictions related to the time series data set "count(Fin)" **2030**. Similarly, lines **2042** and **2044** represent lower and upper confidence levels (95%) for predictions related to the time series data set "Nor" **2032**. Predicted data can additionally or alternatively be presented in a tabular form **2046**.

3.2 Illustrative Time Series Prediction Operations

FIGS. **21-25** illustrate various methods of forecasting data, in accordance with embodiments of the present invention. Although the method **2100** of FIG. **21**, the method **2200** of FIG. **22**, the method **2300** of FIG. **23**, the method **2400** of FIG. **24**, and the method **2500** of FIG. **25** are provided as separate methods, the methods, or aspects thereof, can be combined into a single method or combination of methods. As can be appreciated, additional or alternative steps may also be included in different embodiments.

With initial reference to FIG. **21**, FIG. **21** illustrates a method of computing periodicity utilizing predicted missing values, in accordance with embodiments of the present invention. Such a method may be performed, for example, at a data analysis tool, such as data analysis tool **1916** of FIG. **19**. Initially, at block **2102**, a set of time series data is referenced. The set of time series data might be referenced, for example, from a data store based on an indication of the particular set of time series data in a predict command. In embodiments, a set of time series data determined from raw machine data is received. At block **2104**, a determination is made that the time series data has at least one missing data value. Such a determination may be made, for instance, based on omission of one or more data values within the set of time series data. Subsequently, at block, **2106**, a predicted missing value is generated for each of the at least one missing data values. The predicted missing value for a missing data value can be generated based on a weighted average of a time series data value preceding the missing data value and a time series data value following the missing data value. In embodiments, because several consecutive data values might be missing, it is not necessary that the preceding and following time series data values be immediately adjacent to the missing value. Rather, such time series data values might be the nearest preceding and following time series data values. Further, the value used as the preceding or following time series data may, in some

cases, be a calculation of the value. For instance, assume the data value immediately preceding a missing value is 5 and the value collected prior to that is 4. In such a case, the preceding value may be determined to be an average of the two preceding values, that is 4.5, which can then be used to determining the missing value. The set of time series data and the predicted missing values for each of the at least one missing data values are used to determine periodicity associated with the set of time series data. This is shown at block **2008**.

Turning now to FIG. **22**, FIG. **22** illustrates another method of computing periodicity utilizing predicted missing values, in accordance with embodiments of the present invention. Such a method may be performed, for example, at a data analysis tool, such as data analysis tool **1916** of FIG. **19**. Initially, at block **2202**, a set of time series data is referenced. The set of time series data might be referenced, for example, from a data store based on an indication of the particular set of time series data in a predict command. At block **2204**, a determination is made that the time series data has at least one missing data value. Such a determination may be made, for instance, based on omission of one or more data values within the set of time series data. Subsequently, at block, **2206**, a predicted missing value is generated for each of the at least one missing data values. The predicted missing value for a missing data value can be generated based on a weighted average of a time series data value preceding the missing data value and a time series data value following the missing data value. At block **2208**, a complete set of values is generated by aggregating the observed time series data and the generated predicted missing values. At block **2210**, the complete set of values is used to determine autocorrelations of a plurality of lags associated with the complete set of values. In some embodiments, the number of lags for which autocorrelations are determined can be limited to a threshold number (e.g., **2000**). As can be appreciated, such lags can begin at the first data value and capture a first group of values, begin at a data value that results in the last lag (e.g., lag 2000) being associated with the final value, or anywhere in between. At block **2212**, the greatest or largest autocorrelation is determined to be greater than an autocorrelation threshold. Based on the largest autocorrelation being greater than an autocorrelation threshold, at block **2214**, a lag corresponding with a greatest autocorrelation is identified. Such a lag is then designated as periodicity associated with the complete set of values, as indicated at block **2216**. At block **2218**, the periodicity is used to generate a forecasting model that predicts outcomes expected in the future.

With reference to FIG. **23**, FIG. **23** illustrates a method of predicting expected values, in accordance with embodiments of the present invention. Such a method may be performed, for example, at a data analysis tool, such as data analysis tool **1916** of FIG. **19**. Initially, at block **2302**, a data set is converted to a set of time series data. At block **2304**, a predict command is initiated. Such a predict command may be initiated, for instance, by a user of a client device. Upon initiating the predict command, the set of time series data is read to detect one or more missing values. At block **2306**, predicted missing values are forecasted for the one or more missing values to generate a complete set of data. Predicted missing values can be forecasted using an average of neighboring time series values. At block **2308**, the complete set of data is used to determine a periodicity associated with the complete set of data. The periodicity is used to generate at least one forecasting model that is used to predict outcomes expected in the future, as indicated at block **2310**. Subsequently, at block **2312**, the at least one forecasting model is executed to predict one or more values expected to occur in the future.

Turning to FIG. **24**, FIG. **24** illustrates a method of concurrently predicting expected values, in accordance with embodiments of the present invention. Such a method may be performed, for example, at a client device, such as client device **1904** of FIG. **19**. Initially, at block **2402**, a request for concurrent prediction analysis is received via a graphical user interface, such as a search graphical user interface. In particular, a user may input or select a command to initiate concurrent prediction analysis of multiple time series data sets. Such an input or selection can include an indication of the multiple time series data sets, one or more algorithms to utilize in execute forecasting of expected values for the time series data sets, and/or other parameters used to execute forecasting of expected values for the time series data sets. In response to the request for concurrent prediction analysis, forecasted or predicted values associated with the multiple time series data sets are presented on a display screen of a client device, as indicated at block **2404**. In embodiments, forecasted values associated with each time series data set are concurrently or simultaneously presented on the display screen. For instance, with brief reference to FIG. **20**, forecasted values for the "count(Fin)" time series data set and the "Nor" time series data set are concurrently displayed. In other embodiments, the forecasted values can be independently displayed, for instance, by toggling through forecasted values or by selection of a time series data set for which forecasted values are desired to be viewed.

With reference to FIG. **25**, FIG. **25** illustrates a method of predicting expected values, in accordance with embodiments of the present invention. Such a method may be performed, for example, at a data analysis tool, such as data analysis tool **1916** of FIG. **19**. Initially, at block **2502**, a request for prediction analysis is received. For instance, a user at a client device may input a predict command indicating multiple time series data sets that, when entered, are received at a component performing forecasting or prediction analysis. Although generally described herein as a single request for prediction analysis, as can be appreciated, in some cases, multiple requests for prediction analysis can be communicated. At block **2504**, the request for prediction analysis is parsed to identify an indication of time series data set(s) for which prediction of expected values is desired. At block **2506**, it is determined whether the request includes an indication of multiple time series data sets. If not, the single time series data set specified in the request is processed in accordance with any corresponding parameters to forecast expected values based on the time series data set, as shown at block **2508**. On the other hand, if it is determined that multiple time series data sets are included in the request, at block **2510**, an object is initiated or generated for forecasting data in association with each time series data set. In other words, for each designated time series data set, an object or prediction analysis is separately initiated or generated to independently forecast values expected in light of the corresponding time series data set. At block **2512**, each object or prediction analysis is concurrently executed to determine forecasted values corresponding with the time series data set. In particular, a forecasting algorithm specified or designated for a particular time series data set can be referenced and used to identify one or more expected values. At block **2514**, the forecasted values corresponding with each time series data set are provided, for example, to a client device such as client device **1904** of FIG. **19**. In some cases, the forecasted values generated from each object or prediction

analysis can be combined and provided as a single output. In other cases, the forecasted values generated from each object or prediction analysis can be separately output. The forecasted values can then be presented for display, for instance, via a client device, e.g., such that a user can concurrently or simultaneously view the forecasted values for each of the concurrently processed time series data sets.

3.3 Illustrative Hardware System

The systems and methods described above may be implemented in a number of ways. One such implementation includes computer devices having various electronic components. For example, components of the system in FIG. **19** may, individually or collectively, be implemented with devices having one or more Application Specific Integrated Circuits (ASICs) adapted to perform some or all of the applicable functions in hardware. Alternatively, the functions may be performed by one or more other processing units (or cores), on one or more integrated circuits or processors in programmed computers. In other embodiments, other types of integrated circuits may be used (e.g., Structured/Platform ASICs, Field Programmable Gate Arrays (FPGAs), and other Semi-Custom ICs), which may be programmed in any manner known in the art. The functions of each unit may also be implemented, in whole or in part, with instructions embodied in a memory, formatted to be executed by one or more general or application-specific computer processors.

An example operating environment in which embodiments of the present invention may be implemented is described below in order to provide a general context for various aspects of the present invention. Referring to FIG. **26**, an illustrative operating environment for implementing embodiments of the present invention is shown and designated generally as computing device **2600**. Computing device **2600** is but one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing device **2600** be interpreted as having any dependency or requirement relating to any one or combination of components illustrated.

The invention may be described in the general context of computer code or machine-useable instructions, including computer-executable instructions such as program modules, being executed by a computer or other machine, such as a personal data assistant or other handheld device. Generally, program modules including routines, programs, objects, components, data structures, etc., refer to code that perform particular tasks or implement particular abstract data types. The invention may be practiced in a variety of system configurations, including handheld devices, consumer electronics, general-purpose computers, more specialized computing devices, etc. The invention may also be practiced in distributed computing environments where tasks are performed by remote-processing devices that are linked through a communications network.

With reference to FIG. **26**, computing device **2600** includes a bus **2610** that directly or indirectly couples the following devices: memory **2612**, one or more processors **2614**, one or more presentation components **2616**, input/output (I/O) ports **2618**, I/O components **2620**, and an illustrative power supply **2622**. Bus **2610** represents what may be one or more busses (such as, for example, an address bus, data bus, or combination thereof). Although depicted in FIG. **26**, for the sake of clarity, as delineated boxes that depict groups of devices without overlap between these groups of devices, in reality, this delineation is not so clear cut and a device may well fall within multiple ones of these depicted boxes. For example, one may consider a display to be one of the one or more presentation components **2616** while also being one of the I/O components **2620**. As another example, processors have memory integrated therewith in the form of cache; however, there is no overlap depicted between the one or more processors **2614** and the memory **2612**. A person of skill in the art will readily recognize that such is the nature of the art, and it is reiterated that the diagram of FIG. **26** merely depicts an illustrative computing device that can be used in connection with one or more embodiments of the present invention. It should also be noticed that distinction is not made between such categories as "workstation," "server," "laptop," "handheld device," etc., as all such devices are contemplated to be within the scope of computing device **2600** of FIG. **26** and any other reference to "computing device," unless the context clearly indicates otherwise.

Computing device **2600** typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by computing device **2600** and includes both volatile and nonvolatile media, and removable and non-removable media. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computing device **2600**. Computer storage media does not comprise signals per se, such as, for example, a carrier wave. Communication media typically embodies computer-readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media. Combinations of any of the above should also be included within the scope of computer-readable media.

Memory **2612** includes computer storage media in the form of volatile and/or nonvolatile memory. The memory may be removable, non-removable, or a combination thereof. Typical hardware devices may include, for example, solid-state memory, hard drives, optical-disc drives, etc. Computing device **2600** includes one or more processors **2614** that read data from various entities such as memory **2612** or I/O components **2620**. Presentation component(s) **2616** present data indications to a user or other device. Illustrative presentation components include a display device, speaker, printing component, vibrating component, etc.

I/O ports **2618** allow computing device **2600** to be logically coupled to other devices including I/O components **2620**, some of which may be built in. Illustrative components include a keyboard, mouse, microphone, joystick,

game pad, satellite dish, scanner, printer, wireless device, etc. The I/O components **2620** may provide a natural user interface (NUI) that processes air gestures, voice, or other physiological inputs generated by a user. In some instances, inputs may be transmitted to an appropriate network element for further processing. An NUI may implement any combination of speech recognition, stylus recognition, facial recognition, biometric recognition, gesture recognition both on screen and adjacent to the screen, air gestures, head and eye tracking, and touch recognition (as described elsewhere herein) associated with a display of the computing device **2600**. The computing device **2600** may be equipped with depth cameras, such as stereoscopic camera systems, infrared camera systems, RGB camera systems, touchscreen technology, and combinations of these, for gesture detection and recognition. Additionally, the computing device **2600** may be equipped with accelerometers or gyroscopes that enable detection of motion.

As can be understood, implementations of the present disclosure provide for various approaches to data processing. The present invention has been described in relation to particular embodiments, which are intended in all respects to be illustrative rather than restrictive. Alternative embodiments will become apparent to those of ordinary skill in the art to which the present invention pertains without departing from its scope.

From the foregoing, it will be seen that this invention is one well adapted to attain all the ends and objects set forth above, together with other advantages which are obvious and inherent to the system and method. It will be understood that certain features and subcombinations are of utility and may be employed without reference to other features and subcombinations. This is contemplated by and is within the scope of the claims.

What is claimed is:

1. A computer-implemented method comprising:

receiving a request to perform a predictive analysis in association with multiple time series data sets and a set of forecasting algorithms to forecast values expected to occur in the future based on the corresponding time series data set, each of the multiple time series data sets and the set of forecasting algorithms identified in the request;

concurrently forecasting values expected to occur in the future for each of the multiple time series data sets using the set of forecasting algorithms identified in the received request; and

providing the forecasted values expected to occur in the future for concurrent display with the corresponding time series data set.

2. The computer-implemented method of claim **1** further comprising converting a set of raw data to the multiple time series data sets.

3. The computer-implemented method of claim **1**, wherein the request to perform the predictive analysis comprises a predict command that includes separate indications of each of the multiple time series data sets.

4. The computer-implemented method of claim **1**, wherein the request to perform the predictive analysis comprises an indication of a first time series data set and a first forecasting algorithm to utilize to perform the predictive analysis based on the first time series data set, and an indication of a second time series data set and a second forecasting algorithm to utilize to perform the predictive analysis based on the second time series data set.

5. The computer-implemented method of claim **1**, wherein the request to perform the predictive analysis comprises an indication of a first time series data set, an indication of a second time series data set, and an indication of a forecasting algorithm to utilize to perform the predictive analysis in association with the first time series data set and the second time series data set.

6. The computer-implemented method of claim **1**, wherein the request to perform the predictive analysis comprises an indication of a first time series data set and first corresponding parameters to utilize to perform the predictive analysis in association with the first time series data set, and an indication of a second time series data set and second corresponding parameters to utilize to perform the predictive analysis in association with the second time series data set.

7. The computer-implemented method of claim **1** further comprising parsing the request to identify each of the multiple time series data sets and the set of forecasting algorithms to use in performing the predictive analysis.

8. The computer-implemented method of claim **1** further comprising parsing the request to identify a first forecasting algorithm associated with a first time series data set and a second forecasting algorithm associated with a second time series data set.

9. The computer-implemented method of claim **1**, wherein the concurrent forecasting occurs via executable objects comprising separate and independent instances of a class, and wherein initiating an executable object for each time series data set comprises constructing a different instance of the class for each time series data set.

10. The computer-implemented method of claim **1**, wherein the concurrent forecasting occurs via executable objects, and wherein initiating an executable object for each time series data set comprises creating a different instance of a class for each time series data set and including in each instance of the class a corresponding set of forecasting parameters associated with the corresponding forecasting algorithm.

11. The computer-implemented method of claim **1**, wherein the concurrent forecasting occurs via executable objects executing separately and independently from one another.

12. The computer-implemented method of claim **1**, further comprising: accessing the multiple time series data sets; and concurrently applying the set of forecasting algorithms designated for the corresponding time series data set to generate the forecasted values.

13. The computer-implemented method of claim **1**, further comprising:

accessing the multiple time series data sets; and

concurrently applying the set of forecasting algorithms designated for the corresponding time series data set to generate the forecasted values.

14. The computer-implemented method of claim **1**, wherein the forecasted values are concurrently presented as a graphical visualization in connection with the corresponding time series data sets.

15. The computer-implemented method of claim **1**, wherein forecasted values associated with each of the corresponding time series data sets are concurrently presented in a tabular format.

16. One or more non-transitory computer-readable storage media having instructions stored thereon, wherein the instructions, when executed by a computing device, cause the computing device to:

receiving a request to perform a predictive analysis in association with multiple time series data sets and a set of forecasting algorithms to forecast values expected to occur in the future based on the corresponding time

series data set, each of the multiple time series data sets and the set of forecasting algorithms identified in the request;

concurrently forecasting values expected to occur in the future for each of the multiple time series data sets using the set of forecasting algorithms identified in the received request; and

providing the forecasted values expected to occur in the future for concurrent display with the corresponding time series data set.

17. A computing device comprising:

one or more processors; and

a memory coupled with the one or more processors, the memory having instructions stored thereon, wherein the instructions, when executed by the one or more processors, cause the computing device to:

receiving a request to perform a predictive analysis in association with multiple time series data sets and a set of forecasting algorithms to forecast values expected to occur in the future based on the corresponding time series data set, each of the multiple time series data sets and the set of forecasting algorithms identified in the request;

concurrently forecasting values expected to occur in the future for each of the multiple time series data sets using the set of forecasting algorithms identified in the received request; and

providing the forecasted values expected to occur in the future for concurrent display with the corresponding time series data set.

18. The computing device of claim **17**, wherein the request to perform the predictive analysis comprises a predict command that includes separate indications of each of the multiple time series data sets.

19. The computing device of claim **17**, wherein the request to perform the predictive analysis comprises an indication of a first time series data set and a first forecasting algorithm to utilize to perform the predictive analysis based on the first time series data set, and an indication of a second time series data set and a second forecasting algorithm to utilize to perform the predictive analysis based on the second time series data set.

20. The computing device of claim **17**, wherein the request to perform the predictive analysis comprises an indication of a first time series data set, an indication of a second time series data set, and an indication of a forecasting algorithm to utilize to perform the predictive analysis in association with the first time series data set and the second time series data set.

* * * * *